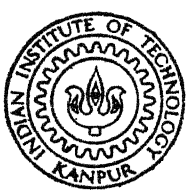
AN ADJACENT EXTREME EFFICIENT POINT PROCEDURE FOR BI-CRITERION LINEAR PROGRAMS

By

V. SUNDARESAN

IMEP
1980
M
SUN
ADJ

TH
IMEP/1980/M
Su 72a



INDUSTRIAL AND MANAGEMENT ENGINEERING PROGRAM

INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

JULY, 1980

AN ADJACENT EXTREME EFFICIENT POINT PROCEDURE FOR BI-CRITERION LINEAR PROGRAMS

A Thesis Submitted
In Partial Fulfilment of the Requirements
for the Degree of
MASTER OF TECHNOLOGY

By
V. SUNDARESAN

to the

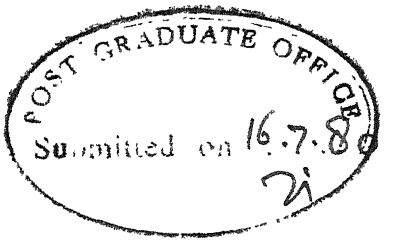
INDUSTRIAL AND MANAGEMENT ENGINEERING PROGRAM
INDIAN INSTITUTE OF TECHNOLOGY, KANPUR

JULY, 1980

IMEP-1980-M-SUN-ADJ

I.I.T. KANPUR
CENTRAL LIBRARY
Acc. No. 63022

8 AUG 1980



CERTIFICATE

This is to certify that the present work on 'An Adjacent Extreme Efficient Point Procedure for Bi-Criterion Linear Programs', by V. Sundaresan has been carried out under my supervision and has not been submitted elsewhere for the award of a degree.

July, 1980

(J.L. Batra)
Professor and Head
Industrial and Management Engg.
Indian Institute of Technology
Kanpur 208016

ACKNOWLEDGEMENTS

I would like to express my deep sense of gratitude to Dr. J.L. Batra for the constant guidance and constructive criticisms, rendered throughout the course of the work. His lively interest and readiness to devote his valuable time on my work has been a constant source of inspiration and has also made this work attain its present stature.

I also wish to express my indebtedness to Dr. S. Sadagopan, who had been of immense help to me in carrying out this work with his valuable suggestions.

Words fail to express my gratitude to my friends for the amusing company they gave which made my stay at IIT Kanpur campus a pleasant and memorable one.

I am also thankful to Mr. J.K. Misra for his excellent typing and Mr. Buddhi Ram Kandiyal for his immaculate cyclostyling work.

V. Sundaresan

CONTENTS

Chapter		Page
I.	INTRODUCTION	1
II.	PROBLEM STATEMENT AND LITERATURE SURVEY	6
	2.1 Problem Statement	6
	2.2 Survey of Literature	7
III.	GENERATION OF EFFICIENT POINTS	11
	3.1 Introduction	11
	3.2 Aneja and Nair's Algorithm	11
	3.2.1 Introduction	11
	3.2.2 Algorithm	12
	3.2.3 Numerical Example	13
	3.3 An Adjacent Extreme Point Procedure	17
	3.3.1 Background	17
	3.3.2 Preliminary Results	18
	3.3.3 Algorithm	22
	3.3.4 Numerical Example	26
IV.	RESULTS AND DISCUSSIONS	30
	4.1 Generation of Random Problems	31
	4.2 Computation Time	31
	4.3 Conclusions	35

Appendix

A. Program Listing for the Aneja and Nair's Algorithm

B. Program Listing for the Proposed Algorithm

C. Program Listing for Generation of Random Problems

SYNOPSIS

Decision Making is the process of selecting an alternative among the available ones. When there are more than one criteria for the selection, the problem is called multi-criteria problem. Most real life problems fall under this category. There are various methods to solve this class of problems. All these methods can be classified into four major catagories depending on the preference information provided by the decision maker and the stage at which it is provided.

In this thesis we develop an algorithm for bicriterion linear programs. The algorithm requires posterior articulation of information. The algorithm generates all efficient extreme points in the objective space. Almost all the existing algorithms for these class of problems generate efficient points in the decision space. However, the algorithm proposed by Aneja and Nair [1] concentrates on the generation of efficient points in the objective space. Hence the proposed algorithm is compared with algorithm proposed by Aneja and Nair. For both the algorithms computer codes in FORTRAN IV were developed and implemented on DEC 1090, Computer system.

For computational simplicity, balanced transportation problems of various sizes were developed. These were solved as linear programs. The size of the linear program varied from problems with 7 constraints 16 variable to 13 constraints 49 variables.

The execution times for the generation of one extreme efficient point for both the algorithms were compared for 24 randomly generated problems of 4 different sizes. It was observed that the proposed algorithms is computationally highly efficient as compared to Aneja and Nair's algorithm. For the smallest size problem, i.e., 4 x 4 balanced transportation problem, the average execution time (average based on 6 problems) per extreme efficient point was 1/6 that of the Aneja and Nair's algorithm. For bigger sized problems, the computational efficiency was even still better.

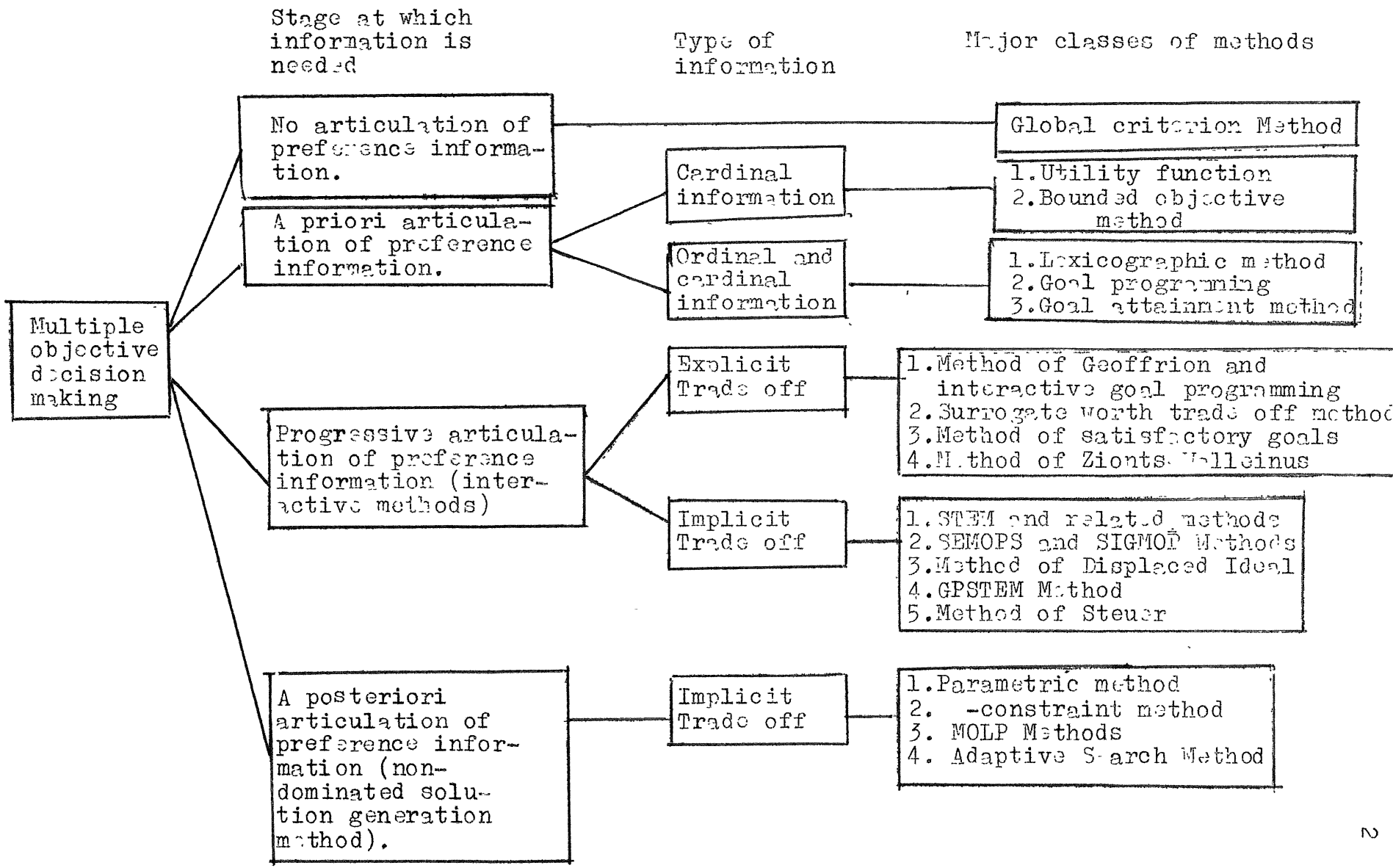
CHAPTER I

INTRODUCTION

Decision making is a process of selecting a possible course of action among the various available ones. There may be one or more criteria which will help justify the selection. If there are more than one criterion then the process is called Multiple Criteria Decision Making (MCDM). Normally the criteria are non-commensurable and conflicting. Of late there has been considerable research activity in the area of multi-criteria decision making since most of the real life problems fall under this category. In the MCDM problems, the best alternative is to be selected keeping in the consideration, the various constraints and criteria. In the process of choosing the best alternative, the decision maker is required to give some preference information. The stage at which it is given is the basis on which the MCDM problems can be classified. The classification of this problems is given in Table 1.1.

The methods for which no articulation of preference information is given, assume that the decision maker will be able to accept the given solution. This assumption is not well justified. The methods where the decision maker is required to give his preference information apriori, suffer from the disadvantage that the decision maker gives this in an information vacuum. The methods which are based

Table 1.1: Classification of Methods for MCDM.



on ordinal preference information given apriori depend heavily on this and the moment the decision maker finds that the information given at an earlier time does not hold any more, the problem is to be solved afresh using the new information.

In the third method, which is classified as interactive method, the decision maker is part of the solution process. There is no need of a priori preference information articulation. The major advantage of this is that decision maker learns about the problem as he proceeds. Since decision maker is part of the solution process, the prospects of the obtained solution being implemented, is bright. But the solution depends upon the accuracy of the local preference the decision maker can indicate. This is one of the disadvantages, these methods suffer from. Another disadvantage is that much more effort is required on the part of the decision maker than other methods.

The last category of methods, listed in Table 1.1 are the methods for which a posteriori artifulation of preference information is given. These methods have an advantage over other methods, because no assumption regarding the decision maker's utility function is required.

The present thesis deals with this class of multi-criteria problems. Solving the multi-criteria problem requires the generation of all non-dominated (efficient) points.

The first generalization of the multi-criteria problem is the bicriterion problems. If all the constraints, and both the criteria are linear, then the problem is called bicriteria linear program. The importance of the bicriterion problems can not be over emphasized. Many situations could be formulated as bicriteria linear programs. For instance in a production system one would like to minimize the cost of production and maximize the production.

A survey of the literature indicates that most of the investigators [2,4,9,10] have approached the bicriteria problem through developing algorithms for generating all the efficient points in the decision space. These algorithms involve the consideration of n dimensional space, where n is the number of decision variables. However, the dimensionality of the solution space can be considerably reduced if instead of the decision space, the objective space is considered. In the present thesis, an algorithm which generates the extreme efficient points in the objective space is developed. The performance of this algorithm is compared with Aneja and Nair's algorithm [1] the only other algorithm reported in the literature which exploits objective space rather than decision space for the generation of extreme efficient points. A set of 24 randomly generated balanced transportation problems of

different sizes are considered for comparing the two algorithms. For both the algorithms, these transportation problems are structured as linear programmes.

The thesis is organized in four chapters. Chapter II deals with problem statement, its formulation and the relevant literature survey. The proposed algorithm is presented in Chapter III. In Chapter IV, a comparison of the proposed algorithm with Aneja and Nair's algorithm is presented. Listing of all the computer programmes developed for this work are given in Appendices A, B and C.

CHAPTER II

PPOBLEM STATEMENT AND LITERATURE SURVEY

2.1 Problem Statement:

Bicriterion Linear Programs (BLP) may be stated as follows:

Consider a Vector Minimum Problem (VMP)

$$\left\{Cx,\ Dx\right\}$$
$$\text{s.t. } Ax \leq b$$
$$x \geq 0 \qquad (1)$$

where x, C and D are n-dimensional vectors, A is a (m x n) matrix and b is m-dimensional vector. C and D are coefficient of the two criterion functions. A represents the technological coefficients of the activities and b is the requirement vector. $Ax \leq b$ represents the constraints of the linear programme (LP). This statement is made without loss of generality, as maximization problems and greater than or equal to constraints can be readily incorporated using standard techniques of LP.

Let us denote the set of feasible solutions by S and its mapping by the two objectives into objective or criterion space by Y. Y is called the pay off set and is defined as

$$Y = \left\{(z_1, z_2) \;\middle|\; z_1 = Cx,\ z_2 = Dx \text{ for some } x \in S\right\}$$

A solution is called superior if it minimizes both the objectives simultaneously. In most real life situations superior solutions do not exist. The objectives are non-commensurable. Under these assumptions the decision maker is interested in efficient solutions or non-dominated solutions. For clarity we define efficient solution before proceeding further.

Definition:

A feasible solution x^o is said to be efficient (non-dominated) if $Cx \leqslant Cx^o$ for some other feasible x implies that $Dx^o \leqslant Dx$ and vice versa.

Essentially it means that no other feasible solution is better with respect to both the criterion. Also by implication $x \in S$ is dominated if there exists atleast one feasible solution x^o such that $Cx^o \leq Cx$ and $Dx^o \leqslant Dx$. Thus, it is always wise to restrict ourselves to non-dominated solution. The solution to (1) can be interpreted as determination of the set of entire efficient points.

2.2 Survey of Literature:

Most of the research in this area has been concentrated on getting all the efficient points in the decision space. One of the early papers in this area is due to Geoffrion [4]. He proposed a scalar maximization approach for obtaining all the efficient solutions. He converted the original problem into the following problem,

Problem P_α:

$$\text{Min. } \alpha Cx + (1-\alpha)\, Dx$$
$$\text{S.t. } Ax \leqslant b$$
$$x \geqslant 0 \qquad (2)$$

where $\alpha \in (0, 1)$. The problem P_α gives all efficient solutions if α is varied in the given range. He proves the necessary and sufficient conditions in the form of a theorem. Further, Geoffrion suggests that his approach can be extended to multi objective programs with more than two criteria. Bacopoulas and Singer [2] adopted a constraint criteria approach to get all the efficient solution in the decision space. One criteria is converted into a constraint at certain level. By parametrically varying this level he proves that all efficient solution can be obtained.

Steuer [9] has given an algorithm which is a modified version of the simplex algorithm. In the column selection rule stage, the efficiency of the next solution is tested by introducing each possible candidate columns one by one. One LP is solved to test the efficiency or non domination of each point. They call this procedure as adjacent efficient basis algorithm.

Yu and Zeleny [10] have given another modification to the simplex procedure. They refer their modified simplex procedure as adjacent non-dominated basis approach. A

modified version of the column selection rule is used to examine non-domination. Essentially they use the $\bar{c}_j$ of each objective to test non-dominance.

Both Steuer and Zeleny and Yu concentrate on generation of efficient points in the decision space, which requires more computing and book keeping. Further, they have addressed to multi criteria problem in general.

Sadagopan [8] has suggested an interactive solution procedure for the bicriteria mathematical programs. Recently, Aneja and Nair [1] have developed an algorithm basically for bicriterion transportation problems. They claim that their algorithm is applicable to bicriterion linear programs also.

Their algorithm generates efficient extreme points in the objective space. The algorithm requires solving of (2K-3) linear programs if there are K (≥ 2) efficient extreme points in the objective space. As the present work considers the same problem and provides algorithm as compared to Aneja and Nair's approach, the details of their algorithm are presented in Chapter III.

The proposed solution methodology involves the generation of efficient extreme points in the criteria space. instead of decision space. This was done for the following reasons. Firstly, the dimensions of the solution space get

reduced if the objective space is considered. For instance in the case of bicriterion problems the objective space is two dimensional while the decision space is n-dimensional where n is the number of decision variables. Secondly, all extreme points in the criteria space correspond to one extreme point in the decision space. Moreover, there may be more than one extreme point in the decision space mapping onto the same extreme point in the objective space. Since the efficient frontier in the objective space is piecewise linear curve, it is enough to restrict ourselves to extreme efficient points. Thus, it is advantageous to look for efficient extreme points in the objective space.

CHAPTER III

GENERATION OF EFFICIENT POINTS

3.1 Introduction:

In the multicriteria problems as already mentioned, one is interested in the generation of efficient points. It was also stated that efficient points in the objective space will be more useful. Because of the polyhedral property of the payoff set, the set of all efficient solution is determined the moment the extreme points of the polyhedron are known. Since the primary motivation to this research is the algorithm proposed by Aneja and Nair[1], this will be discussed first, followed by the algorithm proposed by us.

3.2 Aneja and Nair's Algorithm:

3.2.1 Introduction:

Aneja and Nair proposed the following algorithm for generating the entire set of efficient points in the criteria space. Though the authors have concentrated on the determination of efficient extreme points for the bi-criteria transportation problem, they claim that their algorithm is equally applicable to bi-criteria LP problems. The algorithm initially tests for the existence of a superior solution.

If such a solution does not exist, the linear programming problem is solved using objective functions which are positively weighted average of the objective functions. This process is carried out till all the efficient extreme points are determined.

3.2.2 Algorithm:

Step 0: Find $z_1^{(1)} = \text{Min } (z_1 \mid x \in S)$ and $z_2^{(1)} = \text{Min } (z_2 \mid z_1 = z_1^{(1)}$ and $x \in S)$. Record $(z_1^{(1)}, z_2^{(1)})$ and set k = 1. Similarly, find $z_2^{(2)} = \text{Min } (z_2 \mid x \in S)$ and $z_1^{(2)} = \text{Min } (z_1 \mid z_2 = z_2^{(2)}$ and $x \in S)$. If $(z_1^{(1)}, z_2^{(1)}) = (z_1^{(2)}, z_2^{(2)})$, stop. Else record $(z_1^{(2)}, z_2^{(2)})$ and set k = k+1. Define sets $L = \{(1,2)\}$ and $E = \emptyset$, and go to Step 1.

Step 1: Choose an element $(r, s) \in L$ and set

$$a_1^{(r, s)} = \left| z_2^{(s)} - z_2^{(r)} \right|$$

$$a_2^{(r, s)} = \left| z_1^{(s)} - z_1^{(r)} \right|$$

Let $\bar{x}$ be an optimal solution to the linear program.

$$\text{Min. } a_1^{(r, s)} Cx + a_2^{(r, s)} Dx$$

$$\text{S.t } Ax \leq b$$

$$x \geq 0 \qquad (1)$$

If there are alternate optimal choose an optimal solution $\bar{x}$ for which Cx is minimum.

Let $\bar{z}_1 = Cx$ and $z_2 = Dx$

If $(\bar{z}_1, \bar{z}_2)$ is equal either to $(z_1^{(r)}, z_2^{(r)})$ or $(z_1^{(s)}, z_2^{(s)})$ set $E = E \cup \{(r, s)\}$ and go to Step 2. Otherwise record $(z_1^{(k)}, z_2^{(k)})$ such that $z_1^{(k)} = \bar{z}_1$ and $z_2^{(k)} = \bar{z}_2$, and set $k = k+1$.

$L = L \cup \{(r, k), (k, s)\}$ and goto Step 2.

Step 2: Set $L = L - \{(r, s)\}$. If $L = \emptyset$ stop. Otherwise goto Step 1.

Aneja and Nair have proved the following results.

Theorem 3.1: A point $z^{(k)} = (z_1^{(k)}, z_2^{(k)})$ is an efficient extreme point in the objective space, if and only if $z^{(k)}$ is recorded by the algorithm.

Theorem 3.2: The algorithm is finite and requires exactly $(2k-3)$ iterations when it has $k(\geq 2)$ efficient extreme points.

3.2.3 Numerical Example:

For better understanding of the Aneja and Nair's algorithm, a numerical example considered by them is discussed. Table 3.1 contains the data for a balanced transportation problem with two cost criteria. The two numbers in each cell represent the cost coefficients. The number in the north west corner indicates c_{ij}'s and the numbers in south-east corner denotes d_{ij}'s.

Table 3.1: Data for the problem

				Supply
1 4	2 4	7 3	7 4	8
1 5	9 8	3 9	4 10	19
8 6	9 2	4 5	6 1	17
Demand 11	3	14	16	

Step 0: $z_1^{(1)} = 143\ (= z_1 \mid x \in S)$. The solution to the problem of minimizing Cx absolutely over S is 143.

$$z_2^{(1)} = 265$$

The solution is indicated in Table 3.2. The values given by the side denotes the objective function values.

Table 3.2: Optimal solution with c_{ij} cost coefficients

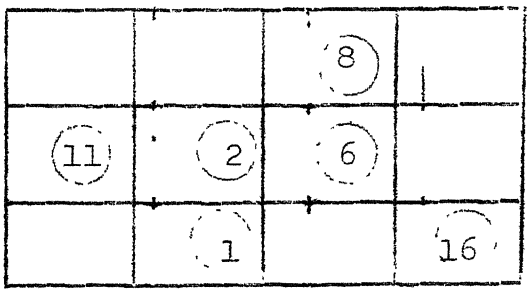
(5)	(3)		
(6)			(13)
		(14)	(3)

$z_1^{(1)} = 143 = Cx$

$z_2^{(1)} = 265 = Dx$

Similarly $z_2^{(2)} = 167$ $(= z_2 \mid x \in S)$. The corresponding solution is indicated in Table 3.3. The objective function values are indicated by the side.

Table 3.3: Optimal solution with d_{ij} costs coefficients.



$z_1^{(2)} = 208$

$z_2^{(2)} = 167$

Iteration No. 1:

Step 1: $(r, s) \in L = (1, 2)$

$a_1^{(r, s)} = 98 \; (= |265 - 167|)$

$a_2^{(r, s)} = 61 \; (= |208 - 143|)$

Min. $a_1^{(r, s)} Cx + a_2^{(r, s)} Dx$

S.t $x \in s$ (2)

The Table 3.4 indicates the modified cost coefficients $a_1^{(r, s)} c_j + a_2^{(r, s)} d_j$ for all $j = 1, \ldots mn$ where m and n are the size of the transportation problem.

Table 3.4: Cost for transportation problem in Iteration No. 1.

					Supply
	238	456	881	946	8
	423	1402	879	1042	19
	1174	1012	717	653	17
Demand	11	3	14	16	

Table 3.5: Solution for problem corresponding to Table 3.4.

(5)	(3)		
(6)		(13)	
		(1)	(16)

$z_1^{(3)} = 156$
$z_2^{(3)} = 200$

Solution to the problem (1) is shown in Table 3.5.

Let, $\bar{z}_1 = C\bar{x} = 156$, $\bar{z}_2 = D\bar{x} = 200$

Since $(\bar{z}_1, \bar{z}_2)$ is not either equal to $(z_1^{(1)}, z_2^{(1)})$ or $(z_1^{(2)}, (z_2^{(2)})$, we have, $(z_1^{(3)}, z_2^{(3)}) = (\bar{z}_1, \bar{z}_2)$.

Now, $k = 3+1 = 4$ and $L = \{(1,2), (1,3), (3,2)\}$.

<u>Step 2</u>: $L = L - \{(1,2)\} = \{(1,3), (3,2)\}$.

<u>Iteration No. 2</u>:

<u>Step 1</u>: $(r,s) = (1, 3)$

Summary of further calculations are shown in the form of Table 3.6.

Table 3.6: Bicriteria Solution in the Objective Space.

Iteration	L	E	Recorded points
1	{(1,2)}	∅	$z^{(1)}$ = (143, 265)
			$z^{(2)}$ = (208, 167)
2	{(1,3),(3,2)}	∅	$z^{(3)}$ = (156, 200)
3	{(3,2)}	{(1,3)}	-
4	{(3,4), (4,2)}	{(1,3)}	$z^{(4)}$ = (176, 175)
5	{(4,2)}	{(1,3), (3,4)}	-
6	{(4,5), (5,2)}	{(1,3), (3,4)}	$z^{(5)}$ = (186, 171)
7	{(5,2)}	{(1,3),(3,4),(4,5)}	-
8	∅	{(1,3),(3,4),(4,5),(5,2)}	-

A computer program was developed to solve the problems by Aneja and Nair's method. The solution shown in Table 3.6 required seven linear program to be solved.

In the following section, the proposed algorithm for generating the efficient extreme points in the objective space is described.

3.3 An Adjacent Extreme Point Procedure:

3.3.1 Background:

In Chapter II, it was pointed out that the determination of efficient extreme points in objective space

rather than the decision space offers two important advantages in case of multicriteria problems. Algorithm described in section 3.2 generates all efficient extreme points in the objective space. Even though the convexity of the efficient frontier is recognized by Aneja and Nair [1], it is not fully exploited. It was also mentioned that 2k-3 linear programs need to be solved if k ($k \geq 2$) efficient extreme points exists.

In this section, an algorithm is developed which attempts to secure an adjacent efficient extreme point in each pivot operation. As explained later, the algorithm may not always be successful in generating the adjacent efficient extreme points. However, this does not destruct the utility of the algorithm since the next extreme efficient point will be secured in few more pivot steps. Before we describe the actual algorithm, certain preliminary results which lay the foundation for the development of the algorithm are stated.

3.3.2 Preliminary Results:

Let us define the original problem as

$$\begin{aligned} \text{PO}: \quad & \text{Minimize} && \{CX,\ DX\} \\ & \text{s.t.} && AX \leq b \\ & && X \geq 0 \end{aligned}$$

where C, and D are n-dimensional vectors representing the cost coefficients, of the two objectives. X is a

n-dimensional vector of variables. A is m x n matrix representing technological coefficients and b is m x 1 vector of requirements.

Consider the following single objective, mathematical programs

P1 : Minimise CX s.t. $AX \leq b$, $X \geq 0$

P2 : Minimise DX s.t. $AX \leq b$, $X \geq 0$

Let the optimal values of P1 and P2 be z_1 and z_2 respectively. Problems P_{z_1} and P_{z_2} are defined as follows:

P_{z_1} : Minimise DX s.t. $AX \leq b$, $CX \leq z_1$, $X \geq 0$

P_{z_2} : Minimise CX s.t. $AX \leq b$, $DX \leq z_2$, $X \geq 0$

Let x_1^* and x_2^* be the optimal solutions for the problems $P_{z_1^*}$ and $P_{z_2^*}$ respectively. Sadagopan [8] has suggested a procedure for reducing the PO problem to P_{z_1} (P_{z_2}) problem with parametrization of $z_1(z_2)$. The procedure utilizes the following theorems.

Theorem 3.3:

In the optimal solution to the program $P_{\bar{z}_2}$ where $\bar{z}_2 \in [z_2^*, Dx_1^*]$ the constraint $DX \leq \bar{z}_2$ will be a binding one i.e. if $\bar{x}$ solve $P_{\bar{z}_2}$ then $D\bar{x} = \bar{z}_2$.

Proof:

Assume the contrary i.e.

$$D\bar{x} < \bar{z}_2 \leq Dx_1^* \tag{3}$$

z_1^* being the absolute minimum of C over S

$$z_1^* = Cx_1^* \leq C\bar{x} \tag{4}$$

Two cases will be considered.

Case 1: Let $Cx_1^* = C\bar{x}$ (5)

(3) and (5) taken together contradict the fact that x_1^* solves $P_{z_1^*}$ since $\bar{x}$ is a better solution.

Case 2: Let $Cx_1^* < C\bar{x}$ (6)

Consider the line segment $[\bar{x}, x_1^*]$ which lies in the convex set S. Over this line segment the linear functions C and D are non-increasing. Because of strict inequality in (6), there exists a feasible solution x' on the line segment $[\bar{x}, x_1^*]$ such that

$$\begin{aligned} D\bar{x} < Dx' < \bar{z}_2 \\ C\bar{x} > Cx' > Cx_1^* \end{aligned} \tag{7}$$

Inequality (7) indicates that x' is a feasible solution to $P_{\bar{z}_2}$ with $Cx' < C\bar{x}$ which contradicts the optimality of $\bar{x}$.

Hence $D\bar{x} = \bar{z}_2$.

Theorem 3.4:

$\bar{x} \in S$ is efficient if and only if $\bar{x}$ solves $P_{\bar{z}_2}$ where $\bar{z}_2 \in [z_2^*, Dx_1^*]$.

The sufficiency and necessity of this theorem is discussed below.

Sufficiency: Let $\bar{x}$ solve $P_{\bar{z}_2}$ (henceforth whenever not mentioned $\bar{z}_2 \in [z_2^*, Dx_1^*]$). If $\bar{x}$ is to be efficient, then it needs to be shown that there does not exist any $x \in S$ such that,

Case (i) $Dx < D\bar{x}$ and $Cx \leq C\bar{x}$

or (8)

Case (ii) $Cx < C\bar{x}$ and $Dx \leq D\bar{x}$

Case (i): Let there be an x ($x \in S$) such that $Dx < D\bar{x} = \bar{z}_2$. Solution x is a feasible solution to $P_{\bar{z}_2}$ but not optimal since the theorem 3.3 states that every optimal solution x^o satisfies $Dx^o = \bar{z}_2$. Hence $Cx > C\bar{x}$. Therefore, x is not efficient.

Case (ii): Consider some $x \in S$ such that $Cx < C\bar{x}$, where $\bar{x}$ solves $P_{\bar{z}_2}$. Since $\bar{x}$ is optimal x cannot be feasible. (otherwise x would have solved $P_{\bar{z}_2}$). This results in $Dx > D\bar{x}$. Consequently x is not efficient.

Thus $\bar{x}$ is efficient.

Necessity: Let E be the set of all efficient solutions. Suppose $\bar{x} \in E$. Obviously $C\bar{x} \leq Cx_2^*$ as otherwise x_2^* will dominate $\bar{x}$. Let $C\bar{x} = \bar{z}_1$. Assume that $\bar{x}$ does not solve $P_{\bar{z}_1}$ but some other $x' \in S$ solves $P_{\bar{z}_1}$. By theorem 3.3, $C\bar{x} = \bar{z}_1 = Cx'$. Since x' is optimal to $P_{\bar{z}_1}$, $Dx' \leq D\bar{x}$. However, because $\bar{x}$ belongs to E this implies that $Dx' \geq D\bar{x}$. Hence $Dx' = D\bar{x}$. Consequently $\bar{x}$ solves $P_{\bar{z}_1}$.

Thus it is possible to generate the entire set of efficient solutions by parametrically solving $P_{\bar{z}_2}$. By theorem 3.3, the generated solution will have specific levels of attainment $(\bar{z}_2)$, of the second criterion. Hadley [5] suggests that for the parametric variation of the right hand side of the constraint set, dual simplex algorithm can be used.

3.3.3 Algorithm (Adjacent Extreme Point Procedure):

Step 0: Solve P1 and P2. Let z_1^* and z_2^* be the optimal solutions to P1 and P2 respectively. Solve $P_{z_1^*}$ and $P_{z_2^*}$. Let x_1^* and x_2^* represent the corresponding optimal solutions.

Step 1: Let $\bar{C}$ and $\bar{D}$ be the relative cost vectors of the objective function of problem P1 whose cost vectors are C and D. (In the first iteration, if alternate optimum to P1 exists then choose among the alternate optimum, that solution which minimizes the second objective with cost vector D). Determine

an adjacent extreme point solution in which the entering variable in x_j such that

$$x_i = \min_j \left\{ -\frac{\bar{c}_j}{\bar{d}_j} \text{ for } \bar{c}_j < 0 \text{ and } \bar{d}_j > 0 \right\} \qquad (9)$$

(In case of tie break them arbitrarily).

Set k = k+1, record $z_1^{(k)} = CX$, $z_2^{(k)} = DX$

If $DX = Dx_2^*$, stop. Otherwise, go to Step 1.

Theorem 3.5:

Algorithm described above generates all efficient extreme points.

Proof: As mentioned in section 3.3.2, the basic idea of this algorithm is to generate all efficient extreme points by solving $P_{\bar{z}_2}$ over $\bar{z}_2 \in [z_2^*, Dx_1^*]$. Only one constraint ($Dx \leq \bar{z}_2$) is added to P1. Hence it can be treated as right hand side parametrization problem for the added constraint. Dual simplex algorithm can be used to do the parametrization.

Let B be the basis matrix associated with the problem P1. The matrix A with objective function in cononical form is represented as follows:

$$\begin{bmatrix} C_B & C_N \\ B & N \end{bmatrix} \begin{bmatrix} x_B \\ x_N \end{bmatrix} = \begin{bmatrix} z \\ b \end{bmatrix} \qquad (10)$$

where B (m x m) and N (m x n-m) are basic and not basic

matrix of A. C is also divided into C_B (1 x n) and C_N (1 x n-m) representing the basic and non-basic cost vectors respectively. With the addition of the constraint $Dx \lesssim \bar{z}_2$, the constraints can be written as follows.

$$\begin{aligned} BX_B + NX_N + 0.s &= b \\ D_B X_B + D_N X_N + 1.s &= \bar{z}_2 \end{aligned} \tag{11}$$

Let A' and b' be denoted as in (12) and (13) respectively. B^* represents the basis of $P_{\bar{z}_2}$. Then

$$A' = \begin{bmatrix} B & N & 0 \\ D_B & D_N & 1 \end{bmatrix} \tag{12}$$

$$b' = \begin{bmatrix} b \\ \bar{z}_2 \end{bmatrix} \tag{13}$$

$$B^* = \begin{bmatrix} B & 0 \\ D_B & 1 \end{bmatrix} \tag{14}$$

Taking the inverse of B^* [6], we have

$$B^{*-1} = \begin{bmatrix} B^{-1} & 0 \\ -D_B B^{-1} & 1 \end{bmatrix} \tag{15}$$

Thus $B^{*-1} A'$ is as follows:

$$\begin{bmatrix} B^{-1} & 0 \\ -D_B B^{-1} & 1 \end{bmatrix} \begin{bmatrix} B & N & 0 \\ D_B & D_N & 1 \end{bmatrix} = \begin{bmatrix} I & B^{-1}N & 0 \\ 0 & D_N - D_B B^{-1} N & 1 \end{bmatrix} \tag{16}$$

where D_B (1xm) and D_N(1 x n-m) are basic and non-basic cost vectors of D respectively.

To determine the right hand side range of the added constraint, the system $B^{*-1} b' \geq 0$ is to be solved. It essentially corresponds to the condition

$$\begin{bmatrix} B^{-1} & 0 \\ -D_B B^{-1} & 1 \end{bmatrix} \begin{bmatrix} b \\ \xi^{(k)} \end{bmatrix} \geq 0 \qquad (17)$$

where $\xi^{(k)}$ represents the range over which $P_{\overline{z}_2}$ continues to represent the optimal basis. Hence the range will be $[\underline{z}_2, \overline{z}_2]$, where $\overline{z}_2$ is the upper limit and $\underline{z}_2$ is the lower limit of $\xi^{(k)}$ by solving the system (17).

The variable to leave the basis will be the slack variable corresponding to the last row and the variable to enter the basis will be decided by the minimum ratio rule of the dual simplex method. Since $D_N - D_B B^{-1} N$ precisely corresponds to the relative cost associated with the objective function with cost coefficients of D vector, the rule to decide the incoming variable in Step 1 of the proposed algorithm corresponds to the minimum ratio rule.

Theorem 3.6:

Algorithm described in Section 3.3.3 terminates in finite number of iterations.

Proof: In dual simplex, no basis is ever repeated. In fact the previous basis becomes infeasible to successive ones. The number of extreme points in the objective space is finite. Hence the algorithm terminates in finite number of iterations.

It should be noted that there may exist tie for the entering variable at some iterations, i.e., more than one variable has the minimum ratio of $-\bar{c}_j/\bar{d}_j$ where $\bar{c}_j$ and $\bar{d}_j$ are dual costs of non-basic variables. The tie can be broken arbitrarily. In such cases, next adjacent extreme efficient point may not be obtained in one pivot operation. In case degeneracy is encountered at some iterations, the lexicographic rule of simplex method can be adopted to prevent cycling.

3.3.4 Numerical Example:

The balanced transportation problem which was given as an illustrated example in Section 3.2.3, is discussed to bring out clearly, the features of the proposed algorithm. Data for the problem is given in Table 3.1.

Solution:

Step 0: Solving P1 and P2, we have $z_1^* = 143$, $z_2^* = 167$. The corresponding solutions appear in Table 3.2 and Table 3.3.

Iteration No. 1:

Step 1: Starting with the solution given in Table 3.2, the dual costs $\bar{c}_{ij}$ and $\bar{d}_{ij}$ of the non-basic cells are

determined. They are shown in Table 3.7.

Table 3.7: Dual costs for the transportation problems at iteration no. 1.

				u_i^1	u_i^2	
(5)	(3)	5 −10	3 −5	0	0	
(6)	7 3	1 −5	(13)	0	1	$z_1^{(1)} = 143$
						$z_2^{(1)} = 265$
5 10	5 6	(14)	(3)	2	−8	
v_j^1	1	2	4	4		
v_j^2	4	4	13	9		

In Table 3.7, the numbers in the north west corner indicates the dual cost $\bar{c}_{ij}$ and the numbers in the south east corner represents the dual cost $\bar{d}_{ij}$. u_i^1 and v_j^1 around the table represents u_i and v_j with respect to c_{ij} costs. u_i^2 and v_j^2 are u_i and v_j with respect to d_{ij} costs.

Cell (2,3) enters the basis in the next iteration since the ratio of $-\bar{c}_{ij}/\bar{d}_{ij}$ is minimum for this non-basic cell.

Now $k = 0+1$, $z_1^{(k)} = 143$, $z_2^{(k)} = 265$.

Since $z_2^{(k)}$ is not equal to 167 (Dx_2^*) go to Step 1.

Iteration No.2:

Step 1: Now the dual costs of non-basic variables are revised. They appear in Table 3.8 with the objective function values by the side.

Table 3.8: Dual costs of non-basic cells at Iteration No.2.

				u_i^1	u_i^2
(5)	(3)	4 −5	2 0	0	0
(6)	7 3	(13)	2 5	0	1
6 5	6 1	(1)	(16)	1	−3

$z_1^{(2)} = 156$

$z_2^{(2)} = 200$

v_j^1	1	2	3	5
v_j^2	4	4	8	4

Now k = k+1 = 2, $z_1^{(2)} = 156$, $z_2^{(2)} = 200$.

Further calculations are shown in Table 3.9.

Table 3.9: Solution to the transportation problem using the proposed algorithm.

Iteration No.	Leaving Variable	Entering Variable	Sl.No. of extreme efficient point (k)	$z^{(k)}$
1	(2,4)	(2,3)	1	(143, 265)
2	(1,1)	(1,3)	2	(156, 200)
3	(3,3)	(3,2)	3	(176, 175)
4	(1,2)	(2,2)	4	(186, 171)
5	-	-	5	(208, 167)

The set of efficient extreme points were generated in 5 iterations. Only two linear programs were solved and the rest of the calculations were only pivot operations. Whereas Aneja and Nair method requires solving of seven linear programs to arrive at the same set of extreme efficient points.

CHAPTER IV

RESULTS AND DISCUSSIONS

Computer codes were developed for the proposed as well as the Aneja and Nair's algorithms. The programmes were written in FORTRAN IV and implemented in DEC 1090 computer system. Since both the algorithms require solving of linear programs, a standard linear program code from International Mathematical and Statistical library was utilized. The listings of programmes for Aneja and Nair's algorithm and the proposed algorithm are given in Appendix A and B respectively. The execution times for the randomly generated problems of various sizes were recorded for both the algorithms along with the number of extreme efficient points generated. It needs to be pointed out that the lexicographic rule required to prevent cycling due to degeneracy was not incorporated in the computer code. This results in repetitive generation of certain points. However, very few points exhibited this characteristics. The tie for entering variable in the proposed algorithm is resolved using the strategy of steeper reduction. This means that of all the non-basic variables, that x_j is chosen to enter the basis for which the product $\bar{c}_j \theta$ is maximum. Here θ represents the level at which x_j enters the basis.

4.1 Generation of Random Problems:

In order to generate feasible and bounded linear programs, balanced transportation problems were generated. The cost coefficients generated were in the interval 1 to 9. The demand and supply were in the interval 1 to 99 for smaller problems and 1 to 15 for bigger problems. A separate Computer programme was developed for the generation of random problems. The programme listing is given in Appendix C.

The randomly generated balanced transportation problem of size m x n, is converted into a linear program with m + n - 1 constraints and mn variables. For example, a 4 x 4 balanced transportation problem results in a linear program with 7 constraints and 16 variables.

4.2 Computation Time:

Computational experience for 24 randomly generated balanced transportation problems was gathered for both the algorithms. Problems of four different sizes viz., 4 x 4, 5 x 5, 6 x 6 and 7 x 7 were generated. Further, six problems for each problem size were considered. The computational times for solving these problems (in case of both the algorithms) are given in Table 4.1. For each problem size, Table 4.2 gives the average execution times for the generation of one efficient point for both the algorithms. The last column of this table gives a ratio measure to compare the computational efficiency of the two algorithms

Acc. No. 63022

Table 4.1: Execution times by the proposed Algorithm for problems examined.

Problem Size		No. of efficient extreme points	Time taken for generation of extreme efficient points by Aneja and Nair's algo. (time in 10^{-3}sec.)	No. of points generated by the proposed algorithm.	Time taken for generation of efficient points (time in 10^{-3}sec.)
4x4	1	8	3397	8	394
	2	2	839	2	344
	3	2	2779	2	381
	4	5	2317	6	384
	5	4	2060	4	379
	6	3	1078	3	283
5x5	1	2	1645	2	665
	2	4	4008	4	770
	3	5	5260	5	810
	4	6	7351	6	925
	5	7	8465	7	869
	6	9	9220	9	808
6x6	1	4	8418	4	1642
	2	4	8181	5	1781
	3	8	17112	8	1866
	4	9	20723	9	2129
	5	10	22685	13	2286
	6	13	30853	14	2388
7x7	1	6	18155	6	2321
	2	7	28897	7	2337
	3	9	30857	11	2588
	4	9	29105	9	2326
	5	11	38964	12	2686
	6	12	42863	14	2957

for the generation of one efficient extreme point. Of the 24 problems considered, for three problems, there was repetitive generation of the same point at some iterations. For one problem of size 6 x 6 there were three repetitions while for two problems of 7 x 7 there were two repetitions each. As mentioned earlier, the lexicographic rule to avoid cycling due to degeneracy was not incorporated in the computer code and this resulted in the repetitive generation of these points. Further whenever a tie existed for the entering variable the next adjacent extreme efficient point was not generated in one pivot operation. In case of tie a point was obtained on the boundary line joining the present extreme point and the next extreme adjacent efficient point. This phenomenon was observed in one problem of size 4 x 4, two problems size 6 x 6 and one problem of problem size 7 x 7. Only one boundary point was generated in each of these problems.

The average execution time for one efficient point is calculated by dividing the total execution time for all the six problems of particular size by the number of efficient extreme points as determined by Aneja and Nair's algorithm. The second and third column of Table 4.2 show the average execution time for one efficient point by Aneja and Nair's algorithm and the proposed algorithm.

Table 4.2: Average execution times for generation of one efficient point by both the algorithms.

No. of problems	Size	Average time for generation of one efficient point by Aneja and Nair's algorithm. (A)	Average time for generation of one efficient point by proposed algorithm (B)	Ratio (A)/(B)
6	4 x 4	445.35	77.32	5.75
6	5 x 5	1089.36	146.88	7.41
6	6 x 6	2219.38	251.91	8.92
6	7 x 7	2777.48	281.38	9.87

From Table 4.2, it is clear that the proposed algorithm is computationally superior to Aneja and Nair's algorithm. For the 4 x 4 problem, the later algorithm required approximately 6 times more execution time. As the problem size increases, the proposed algorithm shows even better computational superiority over the Aneja and Nair's algorithm. Aneja and Nair's algorithm requires solving $2k-3$ linear programs, where k (≥ 2) efficient extreme point exists, whereas the proposed algorithm requires only two linear programs and a little more than k pivot operations. This explains the reason for the computational superiority of the proposed algorithm over the Aneja and Nair's algorithm.

It should be noted that the above stated computer times are based on solving transportation problems using LP code. Further computational time reduction can be envisaged if a good transportation code is utilized, instead of the general linear programming code. This is due to the fact, the transportation algorithm will not require the calculation of the dual cost for all the variables after each pivot operation. Only the u_i's and v_j's along with the corresponding row and column involved in the pivot operation need to be computed again.

4.3 Conclusions:

In this thesis, an algorithm which generates efficient points in the objective space is developed to solve bicriterion linear programs.

The algorithm exploits the convexity of the efficient frontier in the objective space. The original bicriterion problem is reduced to a problem of parametrization using the results of Sadagopan [8]. This resulted in the proposed algorithm to require solving of two linear programs and a little over k pivot operations where k is the number of efficient points. On the other hand, Aneja and Nair's algorithm needs the solution of 2k-3 linear programs as the authors do not fully exploit the convexity of the efficient frontier.

The comparison of the average execution time to generate one efficient extreme point by both the algorithms clearly indicated the computational superiority of the proposed algorithm over Aneja and Nair's algorithm. For instance, the smallest problem i.e., 4 x 4 balanced transportation problem required only one sixth of the average execution time of Aneja and Nair's algorithm. The computational superiority is even more significant when larger problems are considered.

For very large problems number of efficient points may be large. In these cases an interactive procedure could be developed, which incorporates the proposed algorithm, so that the search region of the efficient frontier is reduced.

This algorithm as such can not be extended to problems with more than two criteria. However, there is a need to develop algorithms for generation of efficient points in the objective space for multicriteria problems.

REFERENCES

1. Aneja, Y.P., K.P.K. Nair, 'Bicriteria Transportation Problems', Management Science, 25, 73-78, 1979.

2. Bacopoulos, A and I.Singer, 'On Convex Vectorial Optimization in Linear Spaces', Jl. Opt. Theory and Appl. 21, 175-188, 1977.

3. Ching-Lai Hwang and Abu Syed Md. Masud, 'Multiple Objective Decision Making - Methods and Applications Springer-Verlag, New York, 1979.

4. Geoffrion, A.M., 'Solving Bicriterion Mathematical Programs', Operations Research, 15, 39-54, 1967.

5. Hadley, G.H., Linear Programming , pp. 384, Addison-Wesley, 1978.

6. Hadley, G.H., Linear Programming, pp. 36, Addison-Wesley, 1978.

7. Hadley, G.H., Linear Programming, pp. 422, Addison-Wesley, 1978.

8. Sadagopan, S. and A.Ravindran, 'Interactive Solution of Bicriteria Mathematical Programs' Research Memorandum No. 80-2, School of Industrial Engineering, Purdue University, 1980.

9. Steuer, R.E., 'Linear Multiple Objective Programming: Theory and Computational Experience, Ph.D. Dissertation, University of North Carolina, 1973.

10. Yu, P.L. and M. Zeleny, 'Linear Multiparametric Programming by Multicriteria Simplex Method', Management Science, 23, 159-170, 1976.

APPENDIX A

Program Listing for the Proposed Algorithm

```
00100   C
00200   C*** MAIN ROUTINE
00300           COMMON /B1/M,N,MA(50),MB(50),ICOST1(50,50),ICOST2(50,5
00400           COMMON /B4/A(50,50),B(50),IA,RW(4000),IW(2000)
00500           COMMON /B11/BASIS(50),IBASIS(50)
00600           COMMON /B13/IZCNT,IZ(50,2)
00700           COMMON /B6/X(50,50)
00800           COMMON /IO/INPUT,IOUT
00900           COMMON /IO1/IOUT1,INPUT1
01000           COMMON /B25/ALPHA(5)
01100           LOGICAL END
01200           INTEGER BASIS,DBASIS,TIME1
01300           DATA INPUT,IOUT,IOUT1,INPUT1/20,5,22,23/
01400           DATA IA/50/
01500           END=.FALSE.
01600           NINE=9
01700           OPEN (UNIT=INPUT,FILE='TRANSP.DAT')
01800           OPEN (UNIT=IOUT,FILE='TRANSP.OUT')
01900           OPEN (UNIT=IOUT1,FILE='TIMING.REP',ACCESS='APPEND')
02000           OPEN (UNIT=INPUT1,FILE='TIMING.OLD')
02100   C       TYPE 401
02200   C 401   FORMAT( 'Give the number of problems')
02300   C       ACCEPT *,NOPROB
02400   C       DO 1200 IIJJ=1,NOPROB
02500     1     CALL RTIME(TIME1)
02600           CALL ZERO
02700           CALL DREAD(END)
02800           IF(END) GO TO 1000
02900           CALL SETUP
03000           CALL FIRST
03100           CALL SECOND
03200   C       TYPE *,IZCNT
03300           CALL TIME(TIME1,IZCNT)
03400           CALL PRINT(NINE)
03500           NINE=9
03600           GO TO 1
03700    1000   STOP
03800           CALL UERTST
03900           CALL ZX1LP
04000           END
04100   C
04200   C***   FIRST ***
04300   C
04400           SUBROUTINE FIRST
04500           COMMON /B1/M,N,MA(50),MB(50),ICOST1(50,50),ICOST2(50,.
04600           COMMON /B4/A(50,50),B(50),IA,RW(4000),IW(2000)
04700           COMMON /B6/X(50,50)
04800           COMMON /B13/IZCNT,IZ(50,2)
04900           DIMENSION PSOL(50),DSOL(50),COST(50)
05000           INTEGER ROW,COL
05100           M1=M+N-1;M2=M*N
05200           DO 10 I = 1,M2
05300           PSOL(I)=0.0
05400           DSOL(I)=0.0
05500           COST(I)=0.0
05600           ROW=(I-1)/N+1
05700           COL=I-((ROW-1)*N)
05800           COST(I)=-ICOST1(ROW,COL)
05900     10    CONTINUE
06000           CALL ZX3LP(A,IA,B,COST,M2,0,M1,S,PSOL,DSOL,RW,IW,IER)
06100           DO 15 I = 1,M
06200           DO 15 J = 1,N
06300           IJ=((I-1)*N+J)
06400           A(M1+1,IJ)=-COST(IJ)
06500           COST(IJ)=-ICOST2(I,J)
```

```
06600     15     CONTINUE
06700            B(M1+1)=-S
06800            CALL ZX3LP(A,IA,B,COST,M2,0,M1+1,S,PSOL,DSOL,RW,IW,IER
06900            CALL MATFOM(PSOL)
07000            CALL XCHANG(PSOL,COST,X,M2,SUM)
07100     C      CALL PRINT(2)
07200            TEMP1=0.0
07300            TEMP2=0.0
07400            DO 20 I = 1,M
07500            DO 20 J = 1,N
07600            TEMP1=TEMP1+ICOST1(I,J)*X(I,J)
07700            TEMP2=TEMP2+ICOST2(I,J)*X(I,J)
07800     20     CONTINUE
07900            IZ(1,1)=TEMP1+0.5
08000            IZ(1,2)=TEMP2+0.5
08100            IZCNT=1
08200            CALL PRINT(13)
08300            RETURN
08400            END
08500     C
08600     C*** SECOND ***
08700     C
08800            SUBROUTINE SECOND
08900            COMMON /B1/M,N,MA(50),MB(50),ICOST1(50,50),ICOST2(50,5
09000            COMMON /B4/A(50,50),B(50),IA,RW(4000),IW(2000)
09100            COMMON /B11/BASIS(50),IBASIS(50)
09200            COMMON /B13/IZCNT,IZ(50,2)
09300            COMMON /B6/X(50,50)
09400            INTEGER ENTER,LEAVE
09500            LOGICAL STOP,UBCOND
09600            STOP=.FALSE.
09700            M1=M+N-1;M2=M*N
09800     5      CALL FNDENT(ENTER,STOP)
09900            IF(STOP)GO TO 100
10000            CALL FDLEAV(ENTER,LEAVE,UBCOND)
10100            IF(UBCOND) CALL ERROR(1)
10200            CALL PIVOT(M1,M2,A,B,BASIS,ENTER,LEAVE)
10300            CALL COMPUT
10400            GO TO 5
10500     100    RETURN
10600            END
10700     C
10800            SUBROUTINE PIVOT(M,N,A,B,BASIS,KEYCOL,KEYROW)
10900            DIMENSION A(50,50),B(50),BASIS(50)
11000            INTEGER BASIS
11100            DENOMR=A(KEYROW,KEYCOL)
11200            DO 3 J = 1,N
11300            IF(A(KEYROW,J).EQ.1.0)2,3
11400     2      DO 4 I =1,M
11500            IF(I.EQ.KEYROW)GO TO 4
11600            IF(A(I,J).NE.0.0)GO TO 3
11700     4      CONTINUE
11800            ICHANG=J
11900            GO TO 5
12000     3      CONTINUE
12100            TYPE 505
12200     505    FORMAT(' SOME THING WRONG IN 51290')
12300     5      DO 10 J=1,N
12400            A(KEYROW,J)=A(KEYROW,J)/DENOMR
12500            B(KEYROW)=B(KEYROW)/DENOMR
12600     10     CONTINUE
12700            DO 30 I = 1,M
12800            IF(I.EQ.KEYROW)GO TO 30
12900               DO 20 J=1,N
13000               IF (J.EQ.KEYCOL)GO TO 20
13100               A(I,J)=A(I,J)-A(I,KEYCOL)*A(KEYROW,J)
13200     20        CONTINUE
13300               B(I)=B(I)-A(I,KEYCOL)*B(KEYROW)
13400     30     CONTINUE
13500            DO 35 I =1,M
```

```
13600            A(I,KEYCOL)=0.0
13700            IF(I.EQ.KEYROW)A(KEYROW,KEYCOL)=1.0
13800            IF(BASIS(I).EQ.ICHANG)BASIS(I)=KEYCOL
13900       35   CONTINUE
14000            RETURN
14100            END
14200       C
14300            SUBROUTINE XCHANG(PSOL,COST,X,N,SUM)
14400            COMMON /B1/NOSP,NOSHRT,MA(50),MB(50),ICOST1(50,50),ICOS
14500            DIMENSION PSOL(N),COST(N),X(50,50)
14600            INTEGER ROW,COL
14700            SUM=0.0
14800            DO 30 I = 1,N
14900            ROW=(I-1)/NOSHRT +1
15000            COL=I-((ROW-1)*NOSHRT)
15100            X(ROW,COL)=PSOL(I)
15200            SUM=SUM + PSOL(I)*COST(I)
15300       30   CONTINUE
15400            RETURN
15500            END
15600            SUBROUTINE ZERO
15700            COMMON /B1/M,N,MA(50),MB(50),ICOST1(50,50),ICOST2(50,50
15800            COMMON /B4/A(50,50),B(50),IA,RW(4000),IW(2000)
15900            COMMON /B11/BASIS(50),IBASIS(50)
16000            COMMON /B13/IZCNT,IZ(50,2)
16100            COMMON /B6/X(50,50)
16200            DO 10 I = 1,50
16300            MA(I)=0;MB(I)=0
16400            B(I)=0
16500            BASIS(I)=0
16600            IBASIS(I)=0
16700            IZ(I,1)=0;IZ(I,2)=0
16800            DO 10 J = 1,50
16900            A(I,J)=0.0
17000            X(I,J)=0.0
17100       10   CONTINUE
17200            M=0;N=0;IZCNT=0
17300            DO 15 I = 1,4000
17400            RW(I)=0.0
17500       15   CONTINUE
17600            DO 20 J = 1,2000
17700       20   IW(J)=0
17800            RETURN
17900            END
18000            SUBROUTINE TIME(TIME1,IZCNT)
18100            COMMON /B25/ALPHA(5)
18200            COMMON /IO1/IOUT1,INPUT1
18300            COMMON /B1/M,N,MA(50),MB(50),ICOST1(50,50),ICOST2(50,50
18400            DIMENSION ALPHAO(6)
18500            INTEGER TIME1,TIME2,TIMDIF
18600            CALL RTIME(TIME2)
18700            TIMDIF=TIME2-TIME1
18800       C
18900            READ(INPUT1,140) (ALPHAO(I),I=1,6)
19000       140  FORMAT (15X,6A5)
19100       C
19200            WRITE(IOUT1,150)(ALPHAO(I),I=1,6),IZCNT,TIMDIF
19300       150  FORMAT(6A5,'|',1X,I7,1X,I7,2X)
19400            RETURN
19500            END
19600       C*** MATFOM
19700            SUBROUTINE MATFOM(PSOL)
19800            COMMON /B1/M,N,MA(50),MB(50),ICOST1(50,50),ICOST2(50,5
19900            COMMON /B4/A(50,50),B(50),IA,RW(4000),IW(2000)
20000            COMMON /B11/BASIS(50),DBASIS(50)
20100            DIMENSION BASE(50,50),PSOL(50),DSOL(50)
20200            INTEGER BASIS,IPSOL(50),DBASIS
20300            M1=M+N-1;M2=M*N
20400            DO 10 I = 1,M2
20500       10   IPSOL(I)=PSOL(I)+.005
20600            J=0
20700            DO 20 I = 1,M2
```

```
20600            IF(IPSOL(I).LE.0.0)GO TO 20
20700            J=J+1
20800            BASIS(J)=I
20900      20    CONTINUE
21000            IF(J.EQ.M1)GO TO 40
21100            DO 35 I = 1,M2
21200            DO 25 J = 1,M1
21300            IF(BASIS(J).EQ.I)GO TO 35
21400      25    CONTINUE
21500            BASIS(M1)=I
21600            GO TO 40
21700      35    CONTINUE
21800      40    DO 45 J = 1,M1
21900            DBASIS(J)=BASIS(J)
22000      45    BASE(I,J)=A(I,BASIS(J))
22100            CALL INVERT(BASE,M1)
22200            CALL MATMLT(BASE,A,B,M1,M2)
22300            RETURN
22400            END
22500      C
22600      C
22700      C
22800      C
22900      C
23000      C***
23100      C
23200            SUBROUTINE INVERT (A,N)
23300            INTEGER PIVR,PIVC
23400            DIMENSION A(50,50),PIVR(50),PIVC(50),PIVE(50),Y(50)
23500            DO 4 I=1,N
23600            PIVR(I)=0
23700      4     PIVC(I)=0
23800            K=1
23900      5     PIVE(K)=0.0
24000            DO 9 I=1,N
24100            DO 6 L=1,K
24200            IF (I.EQ.PIVR(L)) GO TO 9
24300      6     CONTINUE
24400            DO 8 J=1,N
24500            DO 7 L=1,K
24600            IF (J.EQ.PIVC(L)) GO TO 8
24700      7     CONTINUE
24800            AB=ABS(A(I,J))
24900            C=ABS(PIVE(K))
25000            IF (AB.LT.C) GO TO 8
25100            II=I
25200            JJ=J
25300            PIVE(K)=A(I,J)
25400      8     CONTINUE
25500      9     CONTINUE
25600            PIVR(K)=II
25700            PIVC(K)=JJ
25800            DO 11 J=1,N
25900      11    A(II,J)=A(II,J)/PIVE(K)
26000            A(II,JJ)=1./PIVE(K)
26100            DO 13 I=1,N
26200            B=A(I,JJ)
26300            IF (I.EQ.II) GO TO 13
26400            A(I,JJ)=-B/PIVE(K)
26500            DO 12 J=1,N
26600            IF (J.EQ.JJ) GO TO 12
26700            A(I,J)=A(I,J)-B*A(II,J)
26800      12    CONTINUE
26900      13    CONTINUE
27000            K=K+1
27100            IF (K.LE.N) GO TO 5
27200            DO 16 J=1,N
27300            DO 15 I=1,N
27400            IR=PIVR(I)
27500            IC=PIVC(I)
```

```
27600     15      Y(IC)=A(IR,J)
27700             DO 16I=1,N
27800     16      A(I,J)=Y(I)
27900             DO 20 I=1,N
28000             DO 19 J=1,N
28100             JR=PIVR(J)
28200             JC=PIVC(J)
28300     19      Y(JR)=A(I,JC)
28400             DO 20 J=1,N
28500     20      A(I,J)=Y(J)
28600             RETURN
28700             END
28800     C
28900     C***
29000     C
29100             SUBROUTINE MATMLT(BASE,A,B,M1,M2)
29200             DIMENSION BASE(50,50),A(50,50),B(50),A1(50,50),B1(50)
29300             DO 30   I = 1,M1
29400             DO 25 J = 1,M2
29500             SUM = 0.0
29600             DO 20 K = 1,M1
29700       20    SUM=SUM+BASE(I,K)*A(K,J)
29800       25    A1(I,J)=SUM
29900       30    CONTINUE
30000             DO 45 I = 1,M1
30100             SUM=0.0
30200             DO 35 J = 1,M1
30300       35    SUM = SUM + BASE(I,J)*B(J)
30400       45    B1(I)=SUM
30500             DO 55 I = 1,M1
30600             B(I)=B1(I)
30700             DO 55 J = 1,M2
30800       55    A(I,J)=A1(I,J)
30900             RETURN
31000             END
31100     C
31200             SUBROUTINE COMPUT
31300             COMMON /B4/A(50,50),B(50),IA,RW(4000),IW(2000)
31400             COMMON /B11/BASIS(50),IBASIS(50)
31500             COMMON /B1/M,N,MA(50),MB(50),ICOST1(50,50),ICOST2(50,50
31600             COMMON /B6/X(50,50)
31700             COMMON /B13/IZCNT,IZ(50,2)
31800             DIMENSION PSOL(50),COST(50)
31900             INTEGER BASIS,DBASIS
32000             M1=M+N-1;M2=M*N
32100             DO 5 I = 1,M2
32200             COST(I)=0.0
32300             PSOL(I)=0.0
32400       5     CONTINUE
32500             DO 15 J = 1,M1
32600       15    PSOL(BASIS(J))=B(J)
32700             CALL XCHANG(PSOL,COST,X,M2,SUM)
32800             IZCNT=IZCNT+1
32900             SUM1=0.0;SUM2=0.0
33000             DO 25 I = 1,M
33100             DO 25 J =1,N
33200             SUM1=SUM1+X(I,J)*ICOST1(I,J)
33300             SUM2=SUM2+X(I,J)*ICOST2(I,J)
33400       25    CONTINUE
33500             IZ(IZCNT,1)=SUM1+0.5
33600             IZ(IZCNT,2)=SUM2+0.5
33700             CALL PRINT(2);CALL PRINT(14)
33800             RETURN
33900             END
34000     C
34100     C*** FNDENT ***
34200     C
34300             SUBROUTINE FNDENT(ENTER,STOP)
34400             COMMON /IO/INPUT,IOUT
34500             COMMON /B1/M,N,MA(50),MB(50),ICOST1(50,50),ICOST2(50,5
```

```
34600            COMMON /B4/A(50,50),B(50),IA,RW(4000),IW(2000)
34700            COMMON /B11/BASIS(50),IBASIS(50)
34800            LOGICAL STOP
34900            DIMENSION DJBAR(50),CJBAR(50)
35000            INTEGER ENTER,ROW,COL,BASIS,DBASIS
35100            STOP=.FALSE.
35200            M1=M+N-1;M2=M*N
35300            ENTER=0
35400            FMIN=9999.
35500            DO 5 I=1,M2
35600            CJBAR(I)=0.0
35700      5     DJBAR(I)=0.0
35800            DO 40 J=1,M2
35900            DO 15 I = 1,M1
36000            IF(J.EQ.BASIS(I))GO TO 40
36100     15     CONTINUE
36200            DUMMY1=0.0;DUMMY2=0.0
36300            DO 20 I =1,M1
36400            ROW=(BASIS(I)-1)/N+1
36500            COL=BASIS(I)-((ROW-1)*N)
36600            DUMMY1=DUMMY1+ICOST1(ROW,COL)*A(I,J)
36700            DUMMY2=DUMMY2+ICOST2(ROW,COL)*A(I,J)
36800     20     CONTINUE
36900            ROW=(J-1)/N+1
37000            COL=J-((ROW-1)*N)
37100            CJBAR(J)=ICOST1(ROW,COL)-DUMMY1
37200            DJBAR(J)=ICOST2(ROW,COL)-DUMMY2
37300            IF(CJBAR(J).GE.0.AND.DJBAR(J).LT.0.0) GO TO 22
37400            GO TO 40
37500     22     IF(FMIN-(-CJBAR(J)/DJBAR(J)))40,28,25
37600     25     ENTER=J
37700            FMIN=-CJBAR(J)/DJBAR(J)
37800     28     XMIN1=9999.0;XMIN2=9999.0
37900            DO 35 I = 1,M1
38000            IF(A(I,ENTER).LE.0.0)GO TO 30
38100            RATIO1=B(I)/A(I,ENTER)
38200            IF(RATIO1.GE.XMIN1)GO TO 30
38300            XMIN1=RATIO1
38400     30     IF(A(I,J).LE.0.0)GO TO 35
38500            RATIO1=B(I)/A(I,J)
38600            IF(RATIO1.GE.XMIN2)GO TO 35
38700            XMIN2=RATIO1
38800     35     CONTINUE
38900            IF(XMIN1.LT.9999.0.AND.XMIN2.LT.9999.0)GO TO 36
39000            GO TO 40
39100     36     MIN=ENTER
39200            IF(CJBAR(J)*XMIN2.GE.DJBAR(ENTER)*XMIN1)MIN=J
39300            ENTER= MIN
39400     40     CONTINUE
39500            WRITE(IOUT,104)(BASIS(I),I=1,M1)
39600     104    FORMAT('BASIS:',20(I5,1X))
39700            WRITE(IOUT,101)(CJBAR(J),J=1,M2)
39800            WRITE(IOUT,102)(DJBAR(J),J=1,M2)
39900     C      DO 40 I = 1,M1
40000     C  40  WRITE(IOUT,103)(A(I,J),J=1,M2),B(I)
40100       101  FORMAT(' CJBAR:',20(F5.1,1X))
40200       102  FORMAT(' DJBAR:',20(F5.1,1X))
40300       103  FORMAT(' A MAT:',20(F5.1,1X))
40400            IF(ENTER.NE.0)RETURN
40500            STOP=.TRUE.
40600            RETURN
40700            END
40800     C
40900     C*** CDLEAV ***
41000     C
41100            SUBROUTINE FDLEAV(ENTER,LEAVE,UBCOND)
41200            COMMON /B1/M,N,MA(50),MB(50),ICOST1(50,50),ICOST2(50,5(
41300            COMMON /B4/A(50,50),B(50),IA,RW(4000),IW(2000)
41400            COMMON /B11/BASIS(50),DBASIS(50)
41500            INTEGER BASIS,DBASIS,ENTER,LEAVE
```

```
41600            LOGICAL UBCOND
41700            XMIN=9999.0;UBCOND=.TRUE.
41800            M1=M+N-1;M2=M*N
41900            DO 10 I = 1,M1
42000            IF(A(I,ENTER).LE.0.0)GO TO 10
42100            RATIO=B(I)/A(I,ENTER)
42200            IF(RATIO.GE.XMIN)GO TO 10
42300            XMIN=RATIO
42400            UBCOND=.FALSE.
42500            LEAVE=I
42600      10    CONTINUE
42700            RETURN
42800            END
42900      C
43000      C***
43100      C
43200            SUBROUTINE ERROR(I)
43300            COMMON /IO/INPUT,IOUT
43400            GO TO (10,15,10) I
43500      10    WRITE(IOUT,101)
43600      101   FORMAT(15X,'INFEASIBLE SOLUTION OR WRONG DATA'//)
43700            GO TO 20
43800      15    WRITE(IOUT,102)
43900      102   FORMAT(15X,'UNBOUNDED SOLUTION'//)
44000      20    X=-1
44100            ROOT=SORT(X)
44200            END
44300      C
44400      C*** SUBROUTINE DREAD ***
44500      C
44600            SUBROUTINE DREAD(END)
44700            LOGICAL END
44800            COMMON /IO/INPUT,IOUT
44900            COMMON /B1/M,N,MA(50),MB(50),ICOST1(50,50),ICOST2(50,50
45000            COMMON /B25/ALPHA(5)
45100            END = .FALSE.
45200            READ(INPUT,112,END=100)(ALPHA(I),I=1,5), M,INTO,N
45300      112   FORMAT(5A5,I4,A2,I4)
45400            READ(INPUT,101) (MA(I),I=1,M)
45500            READ(INPUT,101) (MB(J),J=1,N)
45600            DO 155 J = 1,M
45700      155   READ(INPUT,101)(ICOST1(J,K),K=1,N)
45800            DO 156 J = 1,M
45900      156   READ(INPUT,101)(ICOST2(J,K),K=1,N)
46000      101   FORMAT(40I3)
46100            WRITE(IOUT,113)(ALPHA(I),I=1,5),M,INTO,N
46200      113   FORMAT(//////5A5,I4,A2,I4///)
46300            DO 15 II=1,2
46400            WRITE(IOUT,102) II,(I,I=1,N)
46500            IF(II.EQ.1)WRITE(IOUT,103) (ICOST1(1,J),J=1,N)
46600            IF(II.EQ.2)WRITE(IOUT,103) (ICOST2(1,J),J=1,N)
46700      102   FORMAT(1H1,30X,'THE INPUT COST MATRIX ',I5,//10X,'DESTN
46800            3,5X,20I5)
46900      103   FORMAT(5X,'SOURCE  1'9X,20I5)
47000            DO 10 I = 2,M
47100            IF(II.EQ.2)WRITE(IOUT,104)I,(ICOST2(I,J),J=1,N)
47200            IF(II.EQ.1)WRITE(IOUT,104)I,(ICOST1(I,J),J=1,N)
47300      104   FORMAT (10X,I4,9X,20I5/)
47400      10    CONTINUE
47500       15   CONTINUE
47600            WRITE(IOUT,106) (MA(I),I=1,M)
47700      106   FORMAT(10X,'SUPPLY' 7X,20I5)
47800      107   FORMAT(10X,'DEMAND',7X,20I5)
47900            WRITE(IOUT,107)(MB(J),J=1,N)
48000            WRITE(IOUT,108)
48100      108   FORMAT(1H1,30X,'Solutions by Aneja and Nair method'//)
48200            RETURN
48300      100   END=.TRUE.
48400            RETURN
48500            END
```

```
48600   C
48700   C***
48800   C
48900           SUBROUTINE SETUP
49000           COMMON /B1/M,N,MA(50),MB(50),ICOST1(50,50),ICOST2(50,50
49100           COMMON /B4/A(50,150),B(50),IA,RW(4000),IW(2000),LPCNT
49200           M1=M+N-1
49300           M2=M*N
49400           DO 10 I = 1,M
49500           B(I)=MA(I)
49600           DO 10 J = 1,N
49700           IJ= (I-1)*N + J
49800      10   A(I,IJ)=1.0
49900           DO 15 I = M+1,M1
50000           B(I)=MB(I-M)
50100           DO 15 J = 1,M
50200           IJ=(J-1)*N + (I-M)
50300      15   A(I,IJ)=1.0
50400           RETURN
50500           END
50600   C
50700   C***
50800   C
50900           SUBROUTINE PRINT(CODE)
51000           COMMON /B6/X(50,50)
51100           COMMON /B5/IZ1(50),IZ2(50),IZCNT,IL(50,2),ILCNT
51200           COMMON /IO/INPUT,IOUT
51300           COMMON /B1/M,N,MA(50),MB(50),ICOST1(50,50),ICOST2(50,50
51400   C       COMMON /B13/IZCNT,IZ(50,2)
51500           COMMON /B4/A(50,150),B(50),IX,RW(4000),IW(2000),LPCNT
51600           COMMON /B14/MINCOC(2),MINCOD(2)
51700           INTEGER CODE
51800           REAL MINCOC,MINCOD
51900           GO TO(10,20,30,40,50,60,70,800,900,1000,1000,1000,1000,
52000      10   WRITE(IOUT,110),IZ1(1),IZ2(1),IZ1(2),IZ2(2)
52100     110   FORMAT(15X,'Superior solution exists'/20X,'Z(1,1)='I5,'
52200           1,I5,' Z(2,1)=',I5,' Z(2,2)=',I5//)
52300           RETURN
52400      20   WRITE(IOUT,120) (X(1,J),J=1,N)
52500           DO 25 I = 2,M
52600           WRITE(IOUT,125) (X(I,J),J=1,N)
52700      25   CONTINUE
52800     120   FORMAT(5X,'The solution matrix follows'//5X,20(F6.1,2X)
52900     125   FORMAT(5X,20(F6.1,2X))
53000           RETURN
53100      30   WRITE(IOUT,130)(X(1,J),J=1,N)
53200           DO 35 I = 2,M
53300           WRITE(IOUT,135) (X(I,J),J=1,N)
53400      35   CONTINUE
53500     130   FORMAT(5x,'The shipping matrix follows'/5X,20(F7.1,2X).
53600     135   FORMAT(5X,20(F7.1,2X))
53700           RETURN
53800      40   WRITE(IOUT,140)
53900     140   FORMAT(10X,'This is not an efficient point'//)
54000           RETURN
54100      50   WRITE(IOUT,150)
54200     150   FORMAT(50X,'THE PROBLEM IS TERMINATED '//)
54300           WRITE(IOUT,245) LPCNT
54400     245   FORMAT(20X,'No of LPs solved:',I6/20X,'~~ ~~ ~~~ ~~~~~~
54500           WRITE(IOUT,250) IZCNT
54600     250   FORMAT(40X,'No, of efficient points are:',I5,/40X,29('_
54700           WRITE(IOUT,255)
54800     255   FORMAT(30X,'Points follow:'/30x,'------ ------'//30x,'
54900           1wo'//)
55000           WRITE(IOUT,256)(IZ1(I),IZ2(I),I=1,IZCNT)
55100     256   FORMAT((29X,I5,5X,I5/))
55200           RETURN
55300      60   WRITE(IOUT,160)IZCNT,IZ1(IZCNT),IZCNT,IZ2(IZCNT)
55400     160   FORMAT(20X,'The efficient points are Z(',I3,',1)= ',I5
55500           15X,'Z(',I3,',2)= ',I5)
```

```
55600            RETURN
55700      70    WRITE(IOUT,170)IZ1(1),IZ2(1),IZ1(2),IZ2(2)
55800     170    FORMAT(15X,'Initial extreme points follow'/20X,'Z(1,1)=
55900            1' Z(1,2)=',I5,' Z(2,1)=',I5,' Z(2,2)=',I5//)
56000            RETURN
56100     800    WRITE(IOUT,805)
56200     805    FORMAT(50X,'Solutions by parametrically varying r,h,sid
56300            WRITE(IOUT,120)(X(1,J),J=1,N)
56400            DO 810 I = 2,M
56500            WRITE(IOUT,125)(X(I,J),J=1,N)
56600     810    CONTINUE
56700            IF(IZCNT==2)GO TO 820
56800            WRITE(IOUT,815) IZ1(IZCNT),IZ2(IZCNT)
56900     815    FORMAT(20X,'MIN C(I,J)*X(I,J)',F10.4,'CORRESPNG D(I,J)*'
57000            1,5X,F10.4)
57100            RETURN
57200     820    WRITE(IOUT,10005)
57300   10005    FORMAT(50X,'The primal is infeasbible'//50x,'The proble
57400            2ted and next problem pursued')
57500            RETURN
57600    1000    WRITE(IOUT,1105)
57700            RETURN
57800     900    WRITE(IOUT,1105)
57900    1105    FORMAT(50X,' WHOLE RANGE COVERED')
58000            RETURN
58100            END
58200   C
58300   C***
58400   C
```

APPENDIX B

Program Listing for Aneja and Nair's Algorithm

```
00100   C
00200   CCC PROGRAM MAIN ROUTINE
00300   CCC
00400   C  **** VARABLE DECLARATION
00500           COMMON /B1/M,N,MA(50),MB(50),ICOST1(50,50),ICOST2(50,50)
00600           COMMON /B4/A(50,150),B(50),IA,RW(4000),IW(2000),LPCNT
00700           COMMON /B5/IZ1(50),IZ2(50),IZCNT,IL(50,2),ILCNT
00800           COMMON /IO/INPUT,IOUT
00900           COMMON /B25/ALPHA(5)
01000           COMMON /IO1/IOUT1
01100           INTEGER TIME1,TIME2,TIMDIF
01200           LOGICAL END,ILEMTY,INFCON,SUPRIR,STOP
01300           DATA INPUT,IOUT,IOUT1/20,21,22/
01400           IA=50
01500           OPEN(UNIT=INPUT,DEVICE=DSK,FILE='TRANSP.DAT')
01600           OPEN(UNIT=IOUT,FILE='TRANSP.OUT')
01700   C       OPEN(UNIT=IOUT,FILE='TRANSP.OUT',ACCESS='APPEND')
01800           OPEN(UNIT=IOUT1,FILE='TIMING.REP',ACCESS='APPEND')
01900   C
02000     1     CALL RTIME(TIME1)
02100           CALL ZERO
02200           CALL DREAD (END)
02300           IF(END) GO TO 1500
02400           CALL FEACHK(INFCON)
02500           IF(INFCON) CALL ERROR(1)
02600   CCC
02700   CCC***  SET UP A MATRIX WITH EQUALITY CONSTRAINTS
02800   CCC
02900           CALL SETUP
03000   CCC
03100   CCC*** FIND INITIAL EXTREME POINTS
03200   CCC
03300           CALL INITAL(SUPRIR)
03400           IF(SUPRIR) 11,10
03500     11    CALL TIME(TIME1,1)
03600           CALL PRINT(1)
03700           GO TO 1
03800   CCC
03900   CCC***   CHOOSE NEXT IL ELEMENT. IF IL IS EMPTY TAKE NEXT PROBLEM
04000   CCC
04100     10    CALL PRINT(7)
04200     100   CALL CHOSIL(ILEMTY,IPOSTN)
04300           IF(ILEMTY) GO TO 1000
04400   CCC
04500   CCC*** FIND THE SOLUTION TO THE TRANSPORTATION PROBLEM WITH THIS
04600   CCC*** IL ELEMENT. IF ALTERNATIVE OPTIMA EXITS CHOOSE MIN SIGMA C * X
04700   CCC
04800           CALL STEP21 (IPOSTN,SUM1,SUM2,IR,IS,STOP)
04900           IF(STOP)GO TO 1
05000   CCC
05100   CCC*** DO THE SECOND PART OF THE SETP 2
05200   CCC*** IF THE SOLUTION IS NOT EFFEICIENT DO NO RECORD
05300   CCC*** ELSE RECORD, DO THIS  BY STEP22
05400   CCC
05500           CALL SETP22(SUM1,SUM2,IR,IS)
05600           CALL SETP3(IR,IS)
05700           GO TO 100
05800    1000   CONTINUE
05900           IFIVE=5
06000           CALL PRINT(IFIVE)
06100           CALL TIME(TIME1,IZCNT)
06200           GO TO 1
06300    1500   STOP
06400           CLOSE(UNIT=INPUT)
```

```
06500            CLOSE(UNIT=IOUT)
06600            CALL UERTST
06700            CALL ZX1LP
06800            END
06900
07000
07100
07200
07300
07400            SUBROUTINE ZERO
07500            COMMON /B1/M,N,MA(50),MB(50),ICOST1(50,50),ICOST2(50,50)
07600            COMMON /B4/A(50,150),B(50),IA,RW(4000),IW(2000),LPCNT
07700            COMMON /B5/IZ1(50),IZ2(50),IZCNT,IL(50,2),ILCNT
07800            COMMON /B11/BASIS(50),IBASIS(50)
07900            COMMON /B13/IZCNTD,IZ(50,2)
08000            COMMON /B6/X(50,50)
08100            DO 10 I = 1,50
08200            MA(I)=0;MB(I)=0
08300            B(I)=0
08400            BASIS(I)=0
08500            IBASIS(I)=0
08600            IZ1(I)=0
08700            IZ2(I)=0
08800            IZ(I,1)=0;IZ(I,2)=0
08900            DO 10 J = 1,50
09000            A(I,J)=0.0
09100            X(I,J)=0.0
09200      10    CONTINUE
09300            M=0;N=0;IZCNT=0
09400            ILCNT=0
09500            IZCNTD=0
09600            LPCNT=0
09700            DO 15 I = 1,4000
09800            RW(I)=0.0
09900      15    CONTINUE
10000            DO 20 J = 1,2000
10100      20    IW(J)=0
10200            RETURN
10300            END
10400      C
10500
10600
10700            SUBROUTINE TIME(TIME1,IZCNT)
10800            COMMON /B25/ALPHA(5)
10900            COMMON /IO1/IOUT1
11000            COMMON /B1/M,N,MA(50),MB(50),ICOST1(50,50),ICOST2(50,50)
11100            INTEGER TIME1,TIME2,TIMDIF
11200            CALL RTIME(TIME2)
11300            TIMDIF=TIME2-TIME1
11400            WRITE(IOUT1,150)(ALPHA(I),I=1,5),M,N,TIMDIF,IZCNT
11500      150   FORMAT(5A5,2X,I2,'x',I2,2X,I7,1X,I3,20X)
11600            RETURN
11700            END
11800      C
11900      C*** SUBROUTINE DREAD ***
12000      C
12100
12200
12300
12400            SUBROUTINE DREAD(END)
12500            LOGICAL END
12600            COMMON /IO/INPUT,IOUT
12700            COMMON /B1/M,N,MA(50),MB(50),ICOST1(50,50),ICOST2(50,50)
12800            COMMON /B25/ALPHA(5)
12900            END = .FALSE.
13000            READ(INPUT,112,END=100)(ALPHA(I),I=1,5), M,INTO,N
13100      112   FORMAT(5A5,I4,A2,I4)
13200            READ(INPUT,101) (MA(I),I=1,M)
13300            READ(INPUT,101) (MB(J),J=1,N)
13400            DO 155 J = 1,M
```

```
13500      155    READ(INPUT,101)(ICOST1(J,K),K=1,N)
13600             DO 156 J = 1,M
13700      156    READ(INPUT,101)(ICOST2(J,K),K=1,N)
13800      101    FORMAT(40I3)
13900             WRITE(IOUT,113)(ALPHA(I),I=1,5),M,INTO,N
14000      113    FORMAT(//////5A5,I4,A2,I4///)
14100             DO 15 II=1,2
14200             WRITE(IOUT,102) II,(I,I=1,N)
14300             IF(II.EQ.1)WRITE(IOUT,103) (ICOST1(1,J),J=1,N)
14400             IF(II.EQ.2)WRITE(IOUT,103) (ICOST2(1,J),J=1,N)
14500      102    FORMAT(1H1,30X,'THE INPUT COST MATRIX ',I5,//10X,'DESTN.->'
14600             3,5X,20I5)
14700      103    FORMAT(5X,'SOURCE  1'9X,20I5)
14800             DO 10 I = 2,M
14900             IF(II.EQ.2)WRITE(IOUT,104)I,(ICOST2(I,J),J=1,N)
15000             IF(II.EQ.1)WRITE(IOUT,104)I,(ICOST1(I,J),J=1,N)
15100      104    FORMAT (10X,I4,9X,20I5/)
15200      10     CONTINUE
15300       15    CONTINUE
15400             WRITE(IOUT,106) (MA(I),I=1,M)
15500      106    FORMAT(10X,'SUPPLY' 7X,20I5)
15600      107    FORMAT(10X,'DEMAND',7X,20I5)
15700             WRITE(IOUT,107)(MB(J),J=1,N)
15800             WRITE(IOUT,108)
15900      108    FORMAT(1H1,30X,'Solutions by Aneja and Nair method'//)
16000             RETURN
16100      100    END=.TRUE.
16200             RETURN
16300             END
16400    C
16500    CC***
16600    C
16700
16800
16900
17000             SUBROUTINE SETUP
17100             COMMON /B1/M,N,MA(50),MB(50),ICOST1(50,50),ICOST2(50,50)
17200             COMMON /B4/A(50,150),B(50),IA,RW(4000),IW(2000),LPCNT
17300             M1=M+N-1
17400             M2=M*N
17500             DO 10 I = 1,M
17600             B(I)=MA(I)
17700             DO 10 J = 1,N
17800             IJ= (I-1)*N + J
17900      10     A(I,IJ)=1.0
18000             DO 15 I = M+1,M1
18100             B(I)=MB(I-M)
18200             DO 15 J = 1,M
18300             IJ=(J-1)*N + (I-M)
18400      15     A(I,IJ)=1.0
18500             RETURN
18600             END
18700    C
18800    CCC***
18900
19000
19100    CCC***
19200    CC
19300
19400
19500
19600             SUBROUTINE INITAL(SUPRIR)
19700             COMMON /B1/M,N,MA(50),MB(50),ICOST1(50,50),ICOST2(50,50)
19800             COMMON /B4/A(50,150),B(50),IA,RW(4000),IW(2000),LPCNT
19900             COMMON /B5/IZ1(50),IZ2(50),IZCNT,IL(50,2),ILCNT
20000             COMMON /B6/X(50,50)
20100             DIMENSION COST(150),PSOL(150),DSOL(150)
20200             LOGICAL SUPRIR
20300             INTEGER TWO
20400             M1=M+N-1
```

```
20500            M2=M*N
20600            DO 100 II=1,2
20700            DO 15 I =1,M
20800            DO 15 J =1,N
20900            IJ=(I-1)*N+J
21000            IF(II.EQ.2)COST(IJ)=-ICOST2(I,J)
21100            IF(II.EQ.1)COST(IJ)=-ICOST1(I,J)
21200     15     CONTINUE
21300            CALL ZX3LP(A,IA,B,COST,M2,0,M1,S,PSOL,DSOL,RW,IW,IER)
21400            LPCNT=LPCNT+1
21500            CALL XCHANG(PSOL,COST,X,M2,TOTCOS)
21600            TWO=2
21700            CALL PRINT(TWO)
21800            IF(II.EQ.1)IZ1(II)=-TOTCOS
21900            IF(II.EQ.2)IZ2(II)=-TOTCOS
22000            B(M1+1)=-TOTCOS
22100            DO 25 I = 1,M
22200            DO 25 J = 1,N
22300            IJ=(I-1)*N +J
22400            A(M1+1,IJ)=ICOST2(I,J)
22500            IF(II.EQ.1)A(M1+1,IJ)=ICOST1(I,J)
22600            COST(IJ)=-ICOST2(I,J)
22700            IF(II.EQ.2) COST(IJ)=-ICOST1(I,J)
22800     25     CONTINUE
22900            LPCNT=LPCNT+1
23000            CALL ZX3LP(A,IA,B,COST,M2,0,M1+1,S,PSOL,DSOL,RW,IW,IER)
23100            CALL XCHANG(PSOL,COST,X,M2,TOTCOS)
23200     28     CONTINUE
23300            IF(II.EQ.1)IZ2(II)=-TOTCOS
23400            IF(II.EQ.2)IZ1(II)=-TOTCOS
23500            TWO = 2
23600            CALL PRINT(TWO)
23700     100    CONTINUE
23800            ILCNT=1
23900            IL(1,1)=1
24000            IL(1,2)=2
24100            IZCNT=2
24200            SUPRIR=.FALSE.
24300            IF(IZ1(1).EQ.IZ1(2).AND.IZ2(1).EQ.IZ2(2)) SUPRIR=.TRUE.
24400            RETURN
24500            END
24600     C
24700     C***
24800     C
24900
25000
25100
25200            SUBROUTINE STEP21(IPOSTN,SUM1,SUM2,IR,IS,STOP)
25300            COMMON /IO/INPUT,IOUT
25400            COMMON /B5/IZ1(50),IZ2(50),IZCNT,IL(50,2),ILCNT
25500            COMMON /B1/M,N,MA(50),MB(50),ICOST1(50,50),ICOST2(50,50)
25600            COMMON /B4/A(50,150),B(50),IA,RW(4000),IW(2000),LPCNT
25700            COMMON /B6/X(50,50)
25800            COMMON /B25/ALPHA(5)
25900            DIMENSION COST(150),PSOL(150),DSOL(150)
26000            LOGICAL STOP
26100            INTEGER THREE ,TWO
26200            STOP = .FALSE.
26300            THREE = 3
26400            TWO=2
26500            IR=IL(IPOSTN,1)
26600            IS=IL(IPOSTN,2)
26700            IA1=ABS(IZ2(IS)-IZ2(IR))
26800            IA2=ABS(IZ1(IS)-IZ1(IR))
26900            DO 10 I =1,M
27000            DO 10 J = 1,N
27100            IJ=(I-1)*N+J
27200            COST(IJ)=IA1*ICOST1(I,J) + IA2*ICOST2(I,J)
27300     10     X(I,J)=COST(IJ)
27400            DO 15 I = 1,M*N
```

```
27500     15      COST(I)=-COST(I)
27600             CALL PRINT(THREE)
27700             M1=M+N-1
27800             M2=M*N
27900             LPCNT=LPCNT+1
28000             CALL ZX3LP(A,IA,B,COST,M2,0,M1,TOTCOS,PSOL,DSOL,RW,IW,IER)
28100     25      DO 35 I = 1,M
28200             DO 35 J = 1,N
28300             IJ=(I-1)*N+J
28400             A(M+N,IJ)=IA1*ICOST1(I,J) + IA2*ICOST2(I,J)
28500     35      COST (IJ)=-ICOST1(I,J)
28600             B(M+N)=-TOTCOS
28700             LPCNT=LPCNT+1
28800             CALL ZX3LP(A,IA,B,COST,M2,0,M1+1,S,PSOL,DSOL,RW,IW,IER)
28900             IF(IER.EQ.133)GO TO 55
29000     30      SUM1=0.0;SUM2=0.0
29100             CALL XCHANG(PSOL,COST,X,M2,TOTCOS)
29200     C       CALL PRINT(TWO)
29300             DO 50 I =1,M
29400             DO 50 J = 1,N
29500             SUM1=SUM1+ICOST1(I,J)*X(I,J)
29600     50      SUM2=SUM2+ICOST2(I,J)*X(I,J)
29700             RETURN
29800     55      STOP=.TRUE.
29900             TYPE 601,(ALPHA(III),III=1,5)
30000    601      FORMAT(//5A5,' IS INFEASIBLE'// )
30100             CALL PRINT(14)
30200             RETURN
30300             END
30400     C***
30500     C
30600
30700
30800
30900             SUBROUTINE SETP22 (SUM1,SUM2,IR,IS)
31000             COMMON /B5/IZ1(50),IZ2(50),IZCNT,IL(50,2),ILCNT
31100             COMMON /B6/X(50,50)
31200             REAL SUM1,SUM2
31300             INTEGER FOUR,SIX
31400             FOUR=4;SIX=6
31500             ISUM1=SUM1;ISUM2=SUM2
31600             DIFR1=ABS(SUM1-IZ1(IR))
31700             DIFR2=ABS(SUM2-IZ2(IR))
31800             DIFS1=ABS(SUM1-IZ1(IS))
31900             DIFS2=ABS(SUM2-IZ2(IS))
32000             IF((DIFR1.LT.1.AND.DIFR2.LT.1).OR.(DIFS1.LT.1.AND.DIFS2.LT.1
32100             1))25,20
32200     C   25  CALL PRINT(FOUR)
32300     C*** WRITES THAT THIS POINT IS NOT EFFICIENT POINT
32400       25    RETURN
32500       20    IZCNT=IZCNT+1
32600             IZ1(IZCNT)=ISUM1
32700             IZ2(IZCNT)=ISUM2
32800             ILCNT=ILCNT+1
32900             IL(ILCNT,1)=IR
33000             IL(ILCNT,2)=IZCNT
33100             ILCNT=ILCNT+1
33200             IL(ILCNT,1)=IZCNT
33300             IL(ILCNT,2)=IS
33400     C***  RECORDS THE EFFICIENT POINT
33500             CALL PRINT(2)
33600             CALL PRINT(SIX)
33700             RETURN
33800             END
33900     C
34000     C***
34100     C
34200
34300
34400
```

```
34500            SUBROUTINE SETP3 (IR,IS)
34600            COMMON /B5/IZ1(50),IZ2(50),IZCNT,IL(50,2),ILCNT
34700            DO 10 I =1,ILCNT
34800            IF((IL(I,1).EQ.IR).AND.(IL(I,2).EQ.IS))GO TO 20
34900      10    CONTINUE
35000            RETURN
35100      20    IL(I,1)=0
35200            IL(I,2)=0
35300            RETURN
35400            END
35500  C
35600  CC***
35700  C
35800
35900
36000
36100            SUBROUTINE ERROR(I)
36200            COMMON /IO/INPUT,IOUT
36300            GO TO (10,15,10) I
36400      10    WRITE(IOUT,101)
36500      101   FORMAT(15X,'INFEASIBLE SOLUTION OR WRONG DATA'//)
36600            RETURN
36700      15    WRITE(IOUT,102)
36800      102   FORMAT(15X,'UNBOUNDED SOLUTION'//)
36900            RETURN
37000            END
37100  C
37200  CC***
37300  C
37400
37500
37600
37700            SUBROUTINE PRINT(CODE)
37800            COMMON /B6/X(50,50)
37900            COMMON /B5/IZ1(50),IZ2(50),IZCNT,IL(50,2),ILCNT
38000            COMMON /IO/INPUT,IOUT
38100            COMMON /B1/M,N,MA(50),MB(50),ICOST1(50,50),ICOST2(50,50)
38200  C         COMMON /B13/IZCNT,IZ(50,2)
38300            COMMON /B4/A(50,150),B(50),IX,RW(4000),IW(2000),LPCNT
38400            COMMON /B14/MINCOC(2),MINCOD(2)
38500            INTEGER CODE
38600            REAL MINCOC,MINCOD
38700            GO TO(10,20,30,40,50,60,70,800,900,1000,1000,1000,1000,820)(
38800      10    WRITE(IOUT,110),IZ1(1),IZ2(1),IZ1(2),IZ2(2)
38900      110   FORMAT(15X,'Superior solution exists'/20X,'Z(1,1)='I5,' Z(1,
39000            1,I5,' Z(2,1)=',I5,' Z(2,2)=',I5//)
39100            RETURN
39200      20    WRITE(IOUT,120) (X(1,J),J=1,N)
39300            DO 25 I = 2,M
39400            WRITE(IOUT,125) (X(I,J),J=1,N)
39500      25    CONTINUE
39600      120   FORMAT(5X,'The solution matrix follows'//5X,20(F6.1,2X)/)
39700      125   FORMAT(5X,20(F6.1,2X))
39800            RETURN
39900      30    WRITE(IOUT,130)(X(1,J),J=1,N)
40000            DO 35 I = 2,M
40100            WRITE(IOUT,135) (X(I,J),J=1,N)
40200      35    CONTINUE
40300      130   FORMAT(5x,'The shipping matrix follows'/5X,20(F7.1,2X)//)
40400      135   FORMAT(5X,20(F7.1,2X))
40500            RETURN
40600      40    WRITE(IOUT,140)
40700      140   FORMAT(10X,'This is not an efficient point'//)
40800            RETURN
40900      50    WRITE(IOUT,150)
41000      150   FORMAT(50X,'THE PROBLEM IS TERMINATED '//)
41100            WRITE(IOUT,245) LPCNT
41200      245   FORMAT(20X,'No of LPs solved:',I6/20X,'~~ ~~ ~~~ ~~~~~~'//)
41300            WRITE(IOUT,250) IZCNT
41400      250   FORMAT(40X,'No. of efficient points are:',I5,/40X,29('_')//
```

```
41500           WRITE(IOUT,255)
41600     255   FORMAT(30X,'Points follow:'/30x,'------ ------'//30x,'One',8x,'T
41700           1wo'//)
41800           WRITE(IOUT,256)(IZ1(I),IZ2(I),I=1,IZCNT)
41900     256   FORMAT((29X,I5,5X,I5/))
42000           RETURN
42100     60    WRITE(IOUT,160)IZCNT,IZ1(IZCNT),IZCNT,IZ2(IZCNT)
42200     160   FORMAT(20X,'The efficient points are Z(',I3,',1)= ',I5,
42300           15X,'Z(',I3,',2)= ',I5)
42400           RETURN
42500     70    WRITE(IOUT,170)IZ1(1),IZ2(1),IZ1(2),IZ2(2)
42600     170   FORMAT(15X,'Initial extreme points follow'/20X,'Z(1,1)='I5,
42700           1' Z(1,2)=',I5,' Z(2,1)=',I5,' Z(2,2)=',I5//)
42800           RETURN
42900     800   WRITE(IOUT,805)
43000     805   FORMAT(50X,'Solutions by parametrically varying r.h.side'//)
43100           WRITE(IOUT,120)(X(1,J),J=1,N)
43200           DO 810 I = 2,M
43300           WRITE(IOUT,125)(X(I,J),J=1,N)
43400     810   CONTINUE
43500           IF(IZCNT==2)GO TO 820
43600           WRITE(IOUT,815) IZ1(IZCNT),IZ2(IZCNT)
43700     815   FORMAT(20X,'MIN C(I,J)*X(I,J)',F10.4,'CORRESPNG D(I,J)*X(I,J)'
43800           1,5X,F10.4)
43900           RETURN
44000     820   WRITE(IOUT,10005)
44100   10005   FORMAT(50X,'The primal is infeasbible'//50x,'The problem is abor
44200           2ted and next problem pursued')
44300           RETURN
44400    1000   WRITE(IOUT,1105)
44500           RETURN
44600     900   WRITE(IOUT,1105)
44700    1105   FORMAT(50X,' WHOLE RANGE COVERED')
44800           RETURN
44900           END
45000   C
45100   C***
 5200   C
45300
45400
45500
45600           SUBROUTINE FEACHK(INFCON)
45700           LOGICAL INFCON
45800           COMMON /B1/M,N,MA(50),MB(50),ICOST1(50,50),ICOST2(50,50)
45900           MSHRT=0;MSUP=0
46000           INFCON=.TRUE.
46100           DO 10 I = 1,M
46200           MSUP=MSUP + MA(I)
46300           MSHRT=0
46400           DO 10 J=1,N
46500           IF((MA(I).LT.0).OR.(MB(J).LT.0).OR.(ICOST1(I,J).LT.0).OR.
46600           1(ICOST2(I,J).LT.0))GO TO 15
46700      10   MSHRT = MSHRT + MB(J)
46800           IF(MSHRT.NE.MSUP)  GO TO 15
46900           INFCON=.FALSE.
47000     15    RETURN
47100           END
47200   C
47300   C***
47400   C
47500
47600
47700           SUBROUTINE CHOSIL (ILEMTY,IPOSTN)
47800           COMMON /B5/IZ1(50),IZ2(50),IZCNT,IL(50,2),ILCNT
47900           COMMON /IO/INPUT,IOUT
48000           INTEGER IPOSTN
48100           LOGICAL ILEMTY
48200           ILEMTY=.FALSE.
48300   C       WRITE(IOUT,505),ILCNT
48400     505   FORMAT(' ILCNT=',I5)
```

```
48500            DO 100 I = 1,ILCNT
48600            IF(IL(I,1).GT.0) GO TO 110
48700       100  CONTINUE
48800            ILEMTY=.TRUE.
48900            RETURN
49000       110  IPOSTN=I
49100            RETURN
49200            END
49300  C
49400  C***
49500  C
49600
49700
49800
49900            SUBROUTINE XCHANG(PSOL,COST,X,N,SUM)
50000            COMMON /B1/NOSP,NOSHRT,MA(50),MB(50),ICOST1(50,50),ICOST2(50,50
50100            DIMENSION PSOL(N),COST(N),X(50,50)
50200            INTEGER ROW,COL
50300            SUM=0.0
50400            DO 30 I = 1,N
50500            IP=PSOL(I)+0.5
50600            PSOL(I)=IP
50700            ROW=(I-1)/NOSHRT +1
50800            COL=I-((ROW-1)*NOSHRT)
50900            SUM=SUM + PSOL(I)*COST(I)
51000            X(ROW,COL)=PSOL(I)
51100       30   CONTINUE
51200            RETURN
51300            END
```

APPENDIX C

Program for Generation of Random Problems

```
00100   c With ISEED 2948513, ten problems of 4 by 4 were generated.
00200   c with ISEED 4548511, twenty problems of 5 by 5 were generated.
00300   C with ISEED 2948513, twenty problems of 3 by 3 were generated.
00400          IMPLICIT INTEGER (A-Z)
00500          DIMENSION SUPPLY(50),DEMAND(50),COST1(50,50),COST2(50,50)
00600          REAL R(5000)
00700          NOOFPR=10
00800          IOUT=21
00900          OPEN(UNIT=IOUT,FILE='GENERT.OUT')
01000   C   ISEED given for the first set of problems of 4 by 4 is 2948513
01100   C   ISEED given for the second set of problems of 5 by 5 is 4548511
01200   C   ISEED GIVEN FOR THE FOURTH SET OF PROBLEMS OF 7 BY 7 IS 40205096
01300   C TSEED GIVEN FOR THE FIFTH SET OF PROBLEMS OF 6 BY 6 IS 549504176
01400          ISEED=549504176
01500          NOTIME=1
01600          DO 95 I = 1,NOTIME
01700          IF(I.EQ.1)SIZE = 6
01800          IF(I.NE.1)SIZE=SIZE*2
01900          N=2*(SIZE**2)+2*SIZE
02000          DO 90 JJJ=1,NOOFPR
02100          DO 10 II = 1,N
02200     10   R(II)=0.0
02300          ISEED=ISEED+1
02400          CALL GGUB(ISEED,N,R)
02500          TOTAL=0; RINDEX=0
02600          DO 15 II = 1,SIZE
02700     12   RINDEX=RINDEX+1
02800          INTER=R(RINDEX)*100.0
02900          IF(INTER.EQ.0.OR.INTER.GT.15)GO TO 12
03000          SUPPLY(II)=INTER
03100          TOTAL=TOTAL+SUPPLY(II)
03200     15   CONTINUE
03300   c      PAUSE ' FINISHED SUPPLY'
03400          DO 20 II=1,SIZE
03500          IF(II==SIZE)GO TO 19
03600     16   RINDEX=RINDEX+1
03700          IF(RINDEX==N)200,205
03800          ISEED=ISEED+2
03900    200   CALL GGUB(ISEED,N,R)
04000          RINDEX=0
04100          GO TO 16
04200    205   INTER=R(RINDEX)*100.0
04300          IF((INTER.EQ.0).OR.(INTER.GE.TOTAL).OR.(INTER.GT.15))16,18
04400          TYPE 1949,II,DEMAND(II)
04500   1949   FORMAT(' DEMAND(',I3,')',I4)
04600     18   DEMAND(II)=INTER
04700          TOTAL=TOTAL-INTER
04800          IF(TOTAL .EQ.1)CALL MODIFY(DEMAND,II,TOTAL)
04900          GO TO 20
05000     19   DEMAND(II)=TOTAL
05100     20   CONTINUE
05200   c      PAUSE ' Finished demand'
05300          DO 50 II=1,SIZE
05400          DO 50 JJ=1,SIZE
05500          DO 45 J = 1,2,1
05600     31   RINDEX=RINDEX+1
05700          IF(RINDEX==N)35,40
05800          ISEED=ISEED+2
05900    35    CALL GGUB(ISEED,N,R)
06000          RINDEX=0
06100          GO TO 31
06200     40   INTER=R(RINDEX)*10.0
06300          IF(INTER.EQ.0)GO TO 31
06400          IF(J.EQ.1) COST1(II,JJ)=INTER
06500          IF(J.EQ.2) COST2(II,JJ)=INTER
```

```
06600      45    CONTINUE
06700      50    CONTINUE
06800            WRITE(IOUT,110) JJJ,SIZE,SIZE
06900     110    FORMAT(' Problem no. ',I4,' of size ',I3,' x ',I3)
07000     101    FORMAT(40I3)
07100            WRITE(IOUT,101)(SUPPLY(II),II=1,SIZE)
07200            WRITE(IOUT,101)(DEMAND(II),II=1,SIZE)
07300            DO 82 JN=1,2
07400            DO 82 JP=1,SIZE
07500            IF(JN==1)WRITE(IOUT,101)(COST1(JP,II),II=1,SIZE)
07600            IF(JN==2)WRITE(IOUT,101)(COST2(JP,II),II=1,SIZE)
07700     82     CONTINUE
07800   c        PAUSE ' One problem is over'
07900     90     CONTINUE
08000     95     CONTINUE
08100            STOP
08200            END
08300            SUBROUTINE MODIFY(DEMAND,II,TOTAL)
08400            IMPLICIT INTEGER (A-Z)
08500            DIMENSION DEMAND(50)
08600            MAX=DEMAND(1)
08700            DO 10 I=1,II
08800            IF(DEMAND(I).GE.MAX)  GO TO 5
08900            GO TO 10
09000     5      MAX=DEMAND(I)
09100            INDEX = I
09200     10     CONTINUE
09300            DEMAND(INDEX)=1
09400            TEMP = MAX-DEMAND(INDEX)
09500            TOTAL=TOTAL + TEMP
09600            RETURN
09700            END
```

A63022

A63022

Date Slip

This book is to be returned on the date last stamped.

.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....
.....	.....

CD 6.72.9

IMEP-1880-M-SUN-ADJ